Trion® Interfaces User Guide

UG-TINTF-v7.4
June 2021
www.efinixinc.com

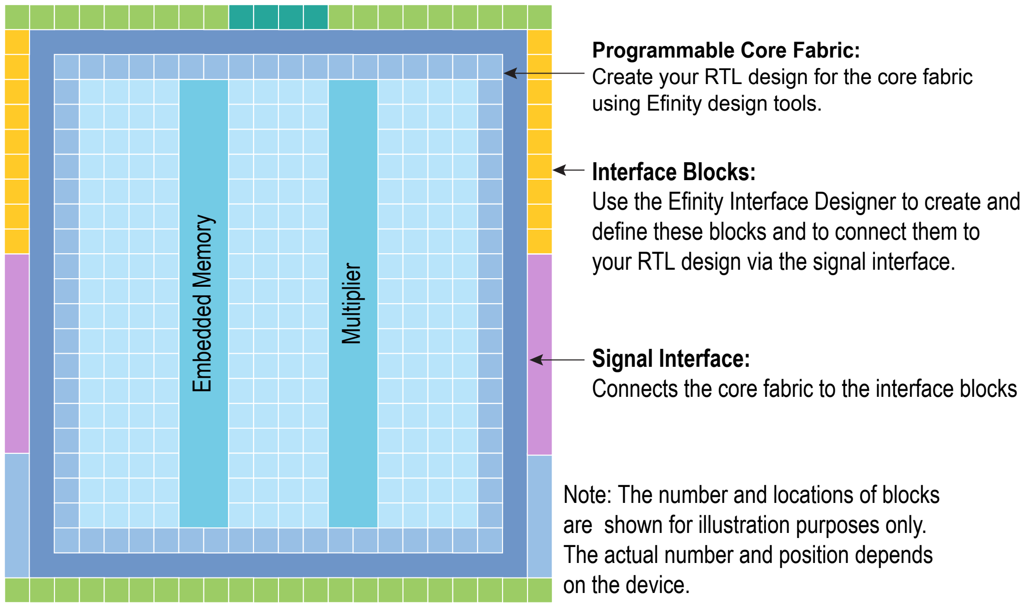
Contents

About the Interface Designer

Trion® FPGAs wrap a Quantum™-accelerated core with a periphery that sends signals out to the device pins. The core contains the logic, embedded memory, and multipliers. The device periphery includes blocks such as GPIO pins, LVDS, MIPI, DDR, and PLLs.

The tools in the Efinity® main window help you design the logic portion of your design. You use the Efinity Interface Designer to build the peripheral portion of your design.

Figure 1: Conceptual View of Interface Blocks


Embedded Memory
Multiplier
Programmable Core Fabric:
Create your RTL design for the core fabric using Efinity design tools.
Interface Blocks:
Use the Efinity Interface Designer to create and define these blocks and to connect them to your RTL design via the signal interface.
Signal Interface:
Connects the core fabric to the interface blocks
Note: The number and locations of blocks are shown for illustration purposes only. The actual number and position depends on the device.

Chapter 1

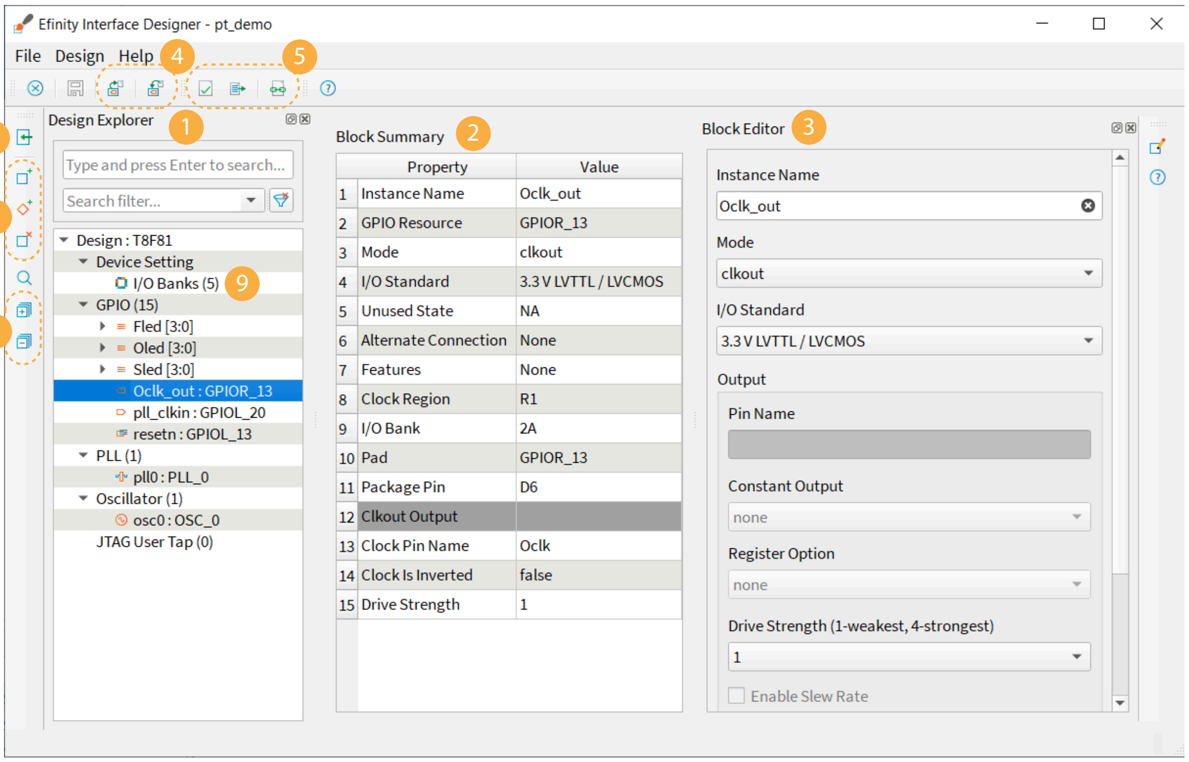
Get Oriented

Contents:

The Interface Designer has four main sections:

- *Design Explorer*—Provides a list view of the interface blocks you have in your design organized by block type. It also includes device-wide settings for the I/O banks and configuration options. Select a block to display it's summary and editor.
- *Block Summary*—Displays the current settings for the selected block.
- *Block Editor*—Provides options and settings for the selected block. The editor may have more than one tab, depending on the block.
- *Resource Assigner*—Provides an easy, tabular method for assigning resources. View by instance (default) or resource.

Figure 2: Interface Designer



Design : T8F81
- Device Setting
  - I/O Banks (5)
- GPIO (15)
  - Fled [3:0]
  - Oled [3:0]
  - Sled [3:0]
  - Oclk_out : GPIOR_13
  - pll_clkin : GPIOL_20
  - resetn : GPIOL_13
- PLL (1)
  - pll0 : PLL_0
- Oscillator (1)
  - osc0 : OSC_0
- JTAG User Tap (0)

	Property	Value
1	Instance Name	Oclk_out
2	GPIO Resource	GPIOR_13
3	Mode	clkout
4	I/O Standard	3.3 V LVTTL / LVCMOS
5	Unused State	NA
6	Alternate Connection	None
7	Features	None
8	Clock Region	R1
9	I/O Bank	2A
10	Pad	GPIOR_13
11	Package Pin	D6
12	Clkout Output	
13	Clock Pin Name	Oclk
14	Clock Is Inverted	false
15	Drive Strength	1

Instance Name: Oclk_out
Mode: clkout
I/O Standard: 3.3 V LVTTL / LVCMOS
Output
Pin Name:
Constant Output: none
Register Option: none
Drive Strength (1-weakest, 4-strongest): 1
Enable Slew Rate

Notes:
1. The Design Explorer shows the interface blocks in your design. They are organized by block type.
2. The block summary shows the settings for the block selected in the Design Explorer.
3. Use the Block Editor to add or change settings for the interface block.
4. You can import or export GPIO resource assignments using a **.csv** or **.isf** file.
5. Use the project management tools to perform design checks, view reports, generate constraints, etc.
6. Click Show/Hide Resource Assigner to toggle a tabular view of assignments.
7. Use the block tools to add or delete blocks and buses.
8. Expand or collapse the Design Explorer folders.
9. The number in parentheses shows the number of used blocks.

When you first open the Interface Designer for your project, the Design Explorer shows the Device Settings folder (with default settings) and empty folders for the interface blocks your chosen device supports. You need to add blocks as required for your design.

Interface Blocks

Trion® FPGAs support a variety of interface blocks. The available blocks differ depending on which FPGA you target. You need to assign a resource for every block you use.

The following table describes the interface blocks supported in the Efinity® software version 2021.1.

Table 1: Trion Interface Blocks

Interface	T4	T8	T13	T20	T35	T55	T85	T120
GPIO	✓	✓	✓	✓	✓	✓	✓	✓
GPIO bus	✓	✓	✓	✓	✓	✓	✓	✓
I/O bank	✓	✓	✓	✓	✓	✓	✓	✓
JTAG User TAP		✓	✓	✓	✓	✓	✓	✓
LVDS		✓	✓	✓	✓	✓	✓	✓
MIPI			✓	✓	✓	✓	✓	✓
DDR				✓	✓	✓	✓	✓
Simple PLL (V1)	✓	✓(1)						
Advanced PLL (V2)		✓(2)	✓	✓	✓	✓	✓	✓
Oscillator	✓	✓						

All interface blocks have an instance name that must be a unique identifier. When you add a new block, the Interface Designer gives the block a unique default name, which you can change.

Note: Many fields in the Block Editor allow arbitrary names. **After you type a new name, press Enter or click Save to save the name.**

Pin names are the top-level ports of the design implemented in the core that connect to the interface block. These names must be legal Verilog HDL or VHDL identifiers.

(1) BGA49 and BGA81 packages.
(2) QFP144 packages.

Package/Interface Support Matrix

Some interfaces are only available in certain packages. The following table describes which interfaces are supported in specific FPGA/package combinations for the Efinity® software v2021.1.

Table 2: Trion Interface/Package Combinations Supported in Efinity® Software v2021.1

Package	T4	T8	T13	T20	T35	T55	T85	T120
F49	∿ ⎍	∿ ⎍						
W80				⎍ ⋗ ▶				
F81	∿ ⎍	∿ ⎍						
QFP144		⎍ ⋗						
F169			⎍ ⋗ ▶	⎍ ⋗ ▶				
F256			⎍ ⋗	⎍ ⋗				
F324				⎍ ⋗ ▶ ▯	⎍ ⋗ ▶ ▯	⎍ ⋗ ▶ ▯	⎍ ⋗ ▶ ▯	⎍ ⋗ ▶ ▯
F400				⎍ ⋗ ▯	⎍ ⋗ ▯			
F484						⎍ ⋗ ▯	⎍ ⋗ ▯	⎍ ⋗ ▯
F576						⎍ ⋗ ▶ ▯	⎍ ⋗ ▶ ▯	⎍ ⋗ ▶ ▯

Trion Family Legend:

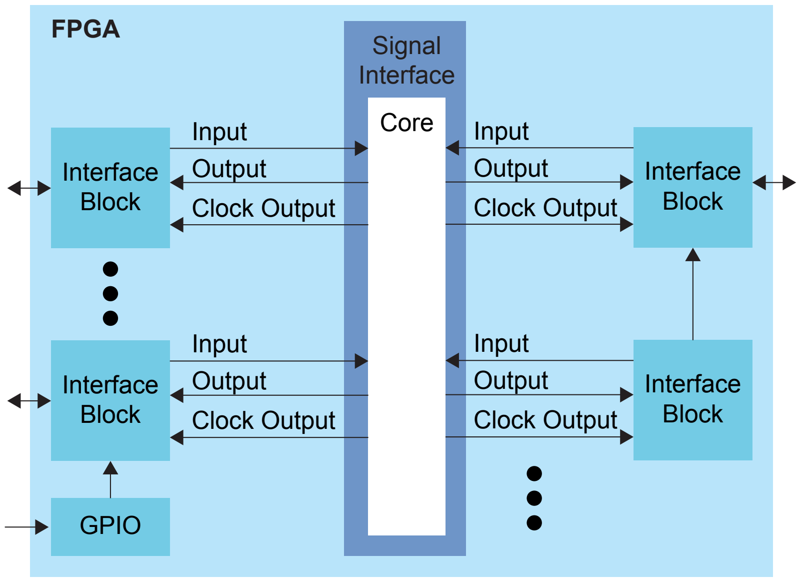
∿ Oscillator	⎍ PLL	⋗ LVDS	▶ MIPI CSI-2 Controller	▯ DDR DRAM Controller

Interface Block Connectivity

The FPGA core fabric connects to the interface blocks through a signal interface. The interface blocks then connect to the package pins. The core connects to the interface blocks using three types of signals:

- *Input*—Input data or clock to the FPGA core
- *Output*—Output from the FPGA core
- *Clock output*—Clock signal from the core clock tree

Figure 3: Interface Block and Core Connectivity



GPIO blocks are a special case because they can operate in several modes. For example, in alternate mode the GPIO signal can bypass the signal interface and directly feed another interface block. So a GPIO configured as an alternate input can be used as a PLL reference clock without going through the signal interface to the core.

When designing for Trion® FPGAs, you create an RTL design for the core and also configure the interface blocks. From the perspective of the core, outputs from the core are inputs to the interface block and inputs to the core are outputs from the interface block.

The Efinity netlist always shows signals from the perspective of the core, so some signals do not appear in the netlist:

- GPIO used as reference clocks are not present in the RTL design, they are only visible in the interface block configuration of the Efinity® Interface Designer.
- The FPGA clock tree is connected to the interface blocks directly. Therefore, clock outputs from the core to the interface are not present in the RTL design, they are only part of the interface configuration (this includes GPIO configured as output clocks).

The following sections describe the different types of interface blocks in the Trion. Signals and block diagrams are shown from the perspective of the interface, not the core.

Designing an Interface

Save	Check Design	Generate Report
Generate Efinity Constraint Files	Exit	

Designing your interface is straightforward: add interface blocks, configure them, and then generate reports and constraints. The Efinity software uses the constraints during compilation to connect signals from the core to your interface.

Note: Refer to Create or Delete a Block on page 10 and Interface Blocks on page 7 for instructions on adding blocks and configuring them.

During the design process, you can generate reports, which are available in the Efinity® Results tab. When you generate reports, the software also saves your design.

Use the design checker to check the interface for errors and to ensure that your settings are valid. The Interface designer displays design issues in the message viewer window. You can also export design issues (**Design > Export Design Issues**) to generate a comma separated values (**.csv**) report to view the issues in a spreadsheet application. When you run the design checker, the software automatically saves your interface.

When you are done configuring your interface, click the Export Efinity Constraints Files button to export the interface constraints to your project. The software saves the design, checks it for errors, generates the interface reports and the interface constraint files.

Click Exit to close the Interface Designer and return to the Efinity® main window.

Note: You can leave the Interface Designer open while running the Efinity® software. However, if you make changes to the Efinity project, the Interface Designer is not updated until the next time you launch it.

Create or Delete a Block

Create Block	Create GPIO Bus	Delete Block

To create a block:

1. Select the folder for the block type you want to create.
2. Click the Create Block button.

To create a GPIO bus, click the GPIO folder and then click the Create GPIO Bus button.

To delete a block, select the block name and click the Delete Block button.

Tip: Right-clicking a folder name opens a context-sensitive menu. From there you can choose **Create Block** (and **Create Bus** for GPIO).

Using the Resource Assigner

Resource Assigner	Switch View	Clear Selected Resource	Clear All Resources
Show/Hide Filter	Clear Filter		

The Resource Assigner provides a tabular view of all GPIO resources in your chosen FPGA and information about them, such as whether they are used, the I/O bank, pad, and package pin, and the instance assigned to the resource.

- The **GPIO: Instance View** shows all GPIO instances in your project.
- The **GPIO: Resource View** shows all GPIO, LVDS, and MIPI RX or TX lane resources and the resources to which you assigned them.

Note: In the Efinity® software v2021.1, you can only view the resources used for LVDS and MIPI lanes in the Resource Assigner. You cannot change or assign resources in this view.

To assign a resource:

1. Open the Resource Assigner by clicking the Show/Hide Resource Assigner button. The software opens to the Instance View, which lists all instances in the design.

   Note: Click Switch View to toggle between instance view and resource view.

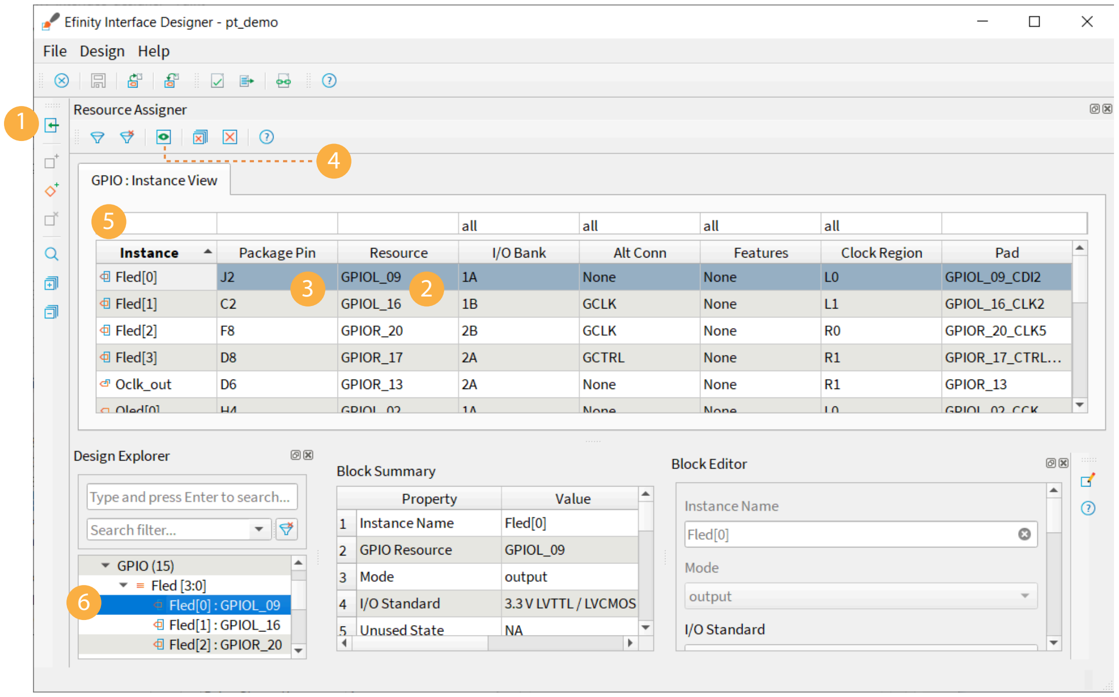
2. In instance view, you can assign pins or resources to the instance. Double-click in the table cell for the item you want to assign. The software displays a drop-down list of available selections.
3. Select an unused resource, instance, or pin.

   Note: If you select a used resource, instance, or pin, the software makes the new assignment, which replaces the previous assignment.

4. Press Enter.

Note: Trion: When using LVDS pins as GPIO, make sure to leave at least 2 pairs of unassigned LVDS pins between any GPIO and LVDS pins in the same bank. This separation reduces noise. The Efinity software issues an error if you do not leave this separation.

Figure 4: Resource Assigner



			all	all	all	all	
Instance	Package Pin	Resource	I/O Bank	Alt Conn	Features	Clock Region	Pad
Fled[0]	J2	GPIOL_09	1A	None	None	L0	GPIOL_09_CDI2
Fled[1]	C2	GPIOL_16	1B	GCLK	None	L1	GPIOL_16_CLK2
Fled[2]	F8	GPIOR_20	2B	GCLK	None	R0	GPIOR_20_CLK5
Fled[3]	D8	GPIOR_17	2A	GCTRL	None	R1	GPIOR_17_CTRL…
Oclk_out	D6	GPIOR_13	2A	None	None	R1	GPIOR_13
Oled[0]	H4	GPIOL_02	1A	None	None	L0	GPIOL_02_CCK

Design Explorer

Type and press Enter to search...

Search filter...

- GPIO (15)
  - Fled [3:0]
    - Fled[0] : GPIOL_09
    - Fled[1] : GPIOL_16
    - Fled[2] : GPIOR_20

Block Summary

	Property	Value
1	Instance Name	Fled[0]
2	GPIO Resource	GPIOL_09
3	Mode	output
4	I/O Standard	3.3 V LVTTL / LVCMOS
5	Unused State	NA

Block Editor

Instance Name: Fled[0]

Mode: output

I/O Standard

Notes:

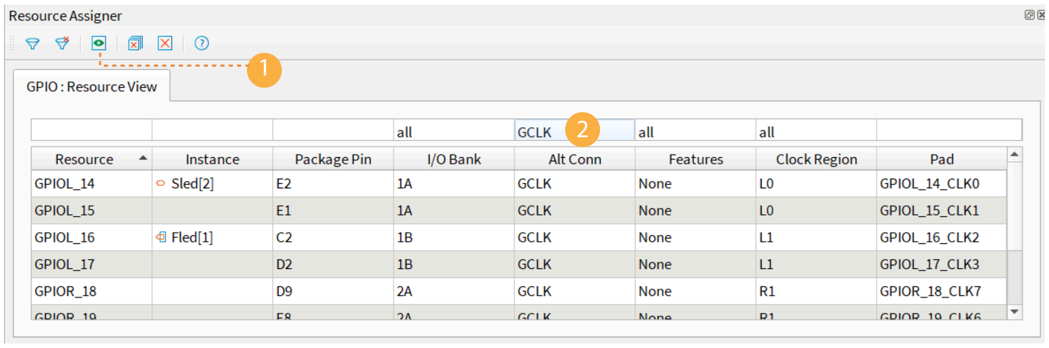
1. Show or hide the Resource Assigner.
2. Double-click in the Resource cell to open the list of available resources.
3. Double-click in the Package Pin cell to open the list of available pins.
4. Click the Switch View button to toggle between Instance View and Resource View.
5. Type in the filter cell above the column you want to filter.
6. Selecting a block in the Design Explorer highlights it in the Resource Assigner.

Resource View

When assigning GPIO, sometimes you want to know which resource can be used as a global clock, global control, or other special function. You can look it up in the pin table for the FPGA and package you are targeting, but an easier way is to use the Resource View in the Resource Assigner.

1. Click the Switch View button to open the Resource View.
2. Double-click in the filter box above the **Alt Conn** column and choose the connection type, for example, **GCTRL**.

Figure 5: Resource View



			all	GCLK	all	all	
Resource	Instance	Package Pin	I/O Bank	Alt Conn	Features	Clock Region	Pad
GPIOL_14	Sled[2]	E2	1A	GCLK	None	L0	GPIOL_14_CLK0
GPIOL_15		E1	1A	GCLK	None	L0	GPIOL_15_CLK1
GPIOL_16	Fled[1]	C2	1B	GCLK	None	L1	GPIOL_16_CLK2
GPIOL_17		D2	1B	GCLK	None	L1	GPIOL_17_CLK3
GPIOR_18		D9	2A	GCLK	None	R1	GPIOR_18_CLK7
GPIOR_19		E8	2A	GCLK	None	R1	GPIOR_19_CLK6

Importing and Exporting Assignments

Although it is nice to use a GUI for adding blocks, in some cases it may be easier to use another format. The Interface Designer lets you import and export assignments using an Interface Scripting File (**.isf**) or comma separated values (**.csv**) file.

When the software reads an imported **.isf**, it processes the entire imported file and shows any issues it found. The import only fails for catastrophic errors. The software:
- Creates new instances defined in the file that do not already exist in the GUI
- Overwrites assignments for existing instances with settings from the file
- Does not delete instances that are in the GUI but were not defined in the file

When the software reads an imported **.csv** file, it compares the imported assignments to the original assignments and reports any issues. If the software finds warnings, it displays them but allows you to finish the import. If it finds errors, it will not finish the import. When importing, the software:
- Deletes instances that you removed
- Creates newly defined instances
- Replaces instances you renamed with the new name

Interface Scripting File

The Interface Scripting File (**.isf**) contains all of the Python API commands to re-create your interface. You can export your design to an **.isf**, manipulate the file, and then re-import it back into the Efinity® software. Additionally, you can write your own **.isf** if desired.

In addition to using the API, you can export and import an **.isf** in the Interface Designer GUI. Click the Import GPIO or Export GPIO buttons and choose **Interface Scripting File (.isf)** under **Format**.

Example: Example Interface Scripting File

```
# Efinity Interface Configuration
# Version: 2020.M.138
# Date: 2020-06-26 14:22
#
# Copyright (C) 2017 - 2020 Efinix Inc. All rights reserved.
#
# Device: T8F81
# Package: 81-ball FBGA (final)
# Project: pt_demo
# Configuration mode: active (x1)
# Timing Model: C2 (final)

# Create instance
design.create_output_gpio("Fled",3,0)
design.create_inout_gpio("Sled",3,0)
design.create_output_gpio("Oled",3,0)
design.create_clockout_gpio("Oclk_out")
design.create_pll_input_clock_gpio("pll_clkin")
design.create_global_control_gpio("resetn")

# Set property, non-defaults
design.set_property("Fled","OUT_REG","REG")
design.set_property("Fled","OUT_CLK_PIN","Fclk")
design.set_property("Sled[0]","IN_PIN","")
design.set_property("Sled[0]","OUT_PIN","Sled[0]")
design.set_property("Sled[1]","IN_PIN","")
design.set_property("Sled[1]","OUT_PIN","Sled[1]")
design.set_property("Sled[2]","IN_PIN","")
design.set_property("Sled[2]","OUT_PIN","Sled[2]")
design.set_property("Sled[3]","IN_PIN","")
design.set_property("Sled[3]","OUT_PIN","Sled[3]")
design.set_property("Oclk_out","OUT_CLK_PIN","Oclk")

# Set resource assignment
design.assign_pkg_pin("Fled[0]","J2")
design.assign_pkg_pin("Fled[1]","C2")
design.assign_pkg_pin("Fled[2]","F8")
design.assign_pkg_pin("Fled[3]","D8")
design.assign_pkg_pin("Sled[0]","E6")
design.assign_pkg_pin("Sled[1]","G4")
design.assign_pkg_pin("Sled[2]","E2")
design.assign_pkg_pin("Sled[3]","G9")
design.assign_pkg_pin("Oled[0]","H4")
design.assign_pkg_pin("Oled[1]","J4")
design.assign_pkg_pin("Oled[2]","A5")
design.assign_pkg_pin("Oled[3]","C5")
design.assign_pkg_pin("Oclk_out","D6")
design.assign_pkg_pin("pll_clkin","C3")
design.assign_pkg_pin("resetn","F1")
```

.csv File for GPIO Blocks

For larger designs with lots of GPIO, it can be simpler to use a spreadsheet application to make assignments. The Resource Assigner allows you to import and export GPIO block assignments using a comma separated values (**.csv**) file. The **.csv** file includes the package pin and pad name, the instance name, and the mode. You can use this method for any type of GPIO, including LVDS pins used as GPIO or HSIO pins used as GPIO.

Table 3: Example GPIO .csv File

Package Pin-Pad Name	Instance Name	Mode
G5-GPIOL_00		
J4-GPIOL_01_SS_N		
H4-GPIOL02_CCK		
G4-GPIOL_03_CDI4	led[0]	output
F4-GPIOL04_CDI0	led[1]	output
J3-GPIOL_05_CDI5	rstn	input
H3-GPIOL_06_CDI1		
...		
(3)	led[6]	inout

When working with the **.csv** file:
- Add your assignments to the **Instance Name** and **Mode** columns.
- Do not modify the package pin-pad names.
- For the mode, specify: input, output, inout, clkout, or none

Note: You cannot make advanced settings such as alternate connections or registering. To make these settings, use the Block Editor.

When the software reads an imported **.csv** file, it performs a comparison between the **.csv** assignments and the original GPIO block assignments and reports any issues. If the software finds warnings, it displays them but allows you to finish the import. If it finds errors, it will not finish the import. When importing, the software:
- Deletes instances that you removed
- Creates newly defined instances
- Replaces instances you renamed with the new name

Interface Designer Output Files

(3) Unassigned instances have a blank field for the Package Pin-Pad Name column.

When you generate constraint files, the Interface Designer creates the following output files. You can view them in the Interface section of the Result pane.

- **<project name>.interface.csv**—Constrains the FPGA design pins used in the interface between the core and the periphery.
- **<project name>.pt.rpt**—Provides information about the interface.
- **<project name>.pinout.csv**—Contains the board design pinout in CSV format.
- **<project name>.pinout.rpt**—Has the board design pinout in a nicely formatted text file format.
- **<project name>.pt_timing.rpt**—Timing report for the Trion® interface logic.
- **<project name>.pt.sdc**—Template SDC file to constrain the FPGA design pins based on the interface configuration.
- **<project name>_template.v**—Template Verilog HDL file defining the FPGA design pins based on the interface configuration.

Scripting an Interface Design

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.[4] Efinix distributes a copy of Python 3 with the Efinity® software to support point tools such as the Debugger and to allow users to write scripts to control compilation.

You use the Efinity® Interface Designer to build the peripheral portion of your design, including GPIO, LVDS, PLLs, MIPI RX and TX lanes, and other hardened blocks. Efinix provides a Python 3 API for the Interface Designer to let you write scripts to control the interface design process. For example, you may want to create a large number of GPIO, or target your design to another board, or export the interface to perform analysis. This user guide describes how to use the API and provides a function reference.

Learn more: Refer to the Python web site, www.python.org/doc, for detailed documentation on the language.

Learn more: For more information on using the Python API to script an interface, refer to the Efinity Interface Designer Python API.

[4] Source: What Is Python? Executive Summary

Chapter 2

Device Settings

Contents:

The Interface Designer has device-wide settings for I/O banks and configuration.

Note: Configuration device-wide settings are not available for T4 and T8 FPGAs.

Configuration Interface

The Configuration device-wide setting lets you control or monitor configuration using the FPGA design implemented in the FPGA core.

Internal Reconfiguration

Efinix® FPGAs (except the T4 and T8) have an internal reconfiguration feature that allows you to control reconfiguration of the FPGA from within the FPGA design. Leave this feature disabled unless you want to use internal reconfiguration.

To enable internal reconfiguration:

1. Click **Device Setting > Configuration**.
2. In the Block Editor, turn on **Enable Internal Reconfiguration Interface**.
3. Indicate the name of the clock pin that will control the internal reconfiguration.
4. Define the FPGA pins that the interface uses.
5. Save.

Note: Refer to AN 010: Using Internal Reconfiguration in Trion FPGAs for instructions on how to use this feature.

I/O Banks Interface

The I/O Banks setting shows the device I/O banks and the I/O voltage each bank uses. Some I/O banks support multiple I/O standards, and you can specify which standard the bank uses. These settings determine the FPGA pinout requirements and timing values of the interface blocks. Some I/O banks can support multiple I/O standards as long as the I/O voltages of the different standards are compatible.

To set the I/O voltage for a bank:

1. Click **Device Setting > I/O Banks**.
2. In the Block Editor, select the I/O voltage for the bank.
   You also select an I/O standard for GPIO blocks. The voltage you select for the I/O bank must be compatible with the settings you choose for any GPIO in this bank.
3. Save.

Note: The I/O banks and their legal configuration are device and package specific. Refer to the data sheet for your chosen FPGA for details on which I/O standards it supports.

I/O Banks

Efinix FPGAs have input/output (I/O) banks for general-purpose usage. Each I/O bank has independent power pins. The number and voltages supported vary by FPGA and package.

Learn more: Refer to the FPGA pinout for information on the I/O bank assignments.

Trion I/O Banks

Table 4: I/O Banks by FPGA and Package

Package	I/O Banks	Voltage (V)	DDIO Support	Merged Banks
T4				
BGA49, BGA81	1A - 1C, 2A, 2B	1.8, 2.5, 3.3	-	-
T8				
BGA49, BGA81	1A - 1C, 2A, 2B	1.8, 2.5, 3.3	-	-
QFP144	1A - 1E, 3A - 3E	1.8, 2.5, 3.3	1B, 1C, 1D, 3B, 3C, 3D, 3E	1C_1D, 3B_3C
	4A, 4B	3.3	-	-
T13				
BGA169	1A - 1E, 3A - 3E	1.8, 2.5, 3.3	1B, 1C, 1D, 3B, 3C, 3D, 3E	1B_1C_1D, 3A_3B, 3C_3D_3E
	4A, 4B	3.3	-	-
BGA256	1A - 1E, 3A - 3E	1.8, 2.5, 3.3	1B, 1C, 1D, 3B, 3C, 3D, 3E	1B_1C, 1D_1E. 3A_3B_3C, 3D_3E
	4A, 4B	3.3	-	-
T20				
WLCSP80	1A-1E, 3A-3E	1.8, 2.5, 3.3	1B, 1D, 3C, 3D, 3E	1B_1C_1D_1E, 3A_3B_3C, 3D_3E_4A_4B
BGA169	1A - 1E, 3A - 3E	1.8, 2.5, 3.3	1B, 1C, 1D, 3B, 3C, 3D, 3E	1B_1C_1D, 3A_3B, 3C_3D_3E
	4A, 4B	3.3	-	-
BGA256	1A - 1E, 3A - 3E	1.8, 2.5, 3.3	1B, 1C, 1D, 3B, 3C, 3D, 3E	1B_1C, 1D_1E. 3A_3B_3C, 3D_3E
	4A, 4B	3.3	-	-
BGA324	1A - 1E, 2A - 2C, 3A - 3C, 4A, 4B, TR, BR	1.8, 2.5, 3.3	1A - 1E, 3C, TR, BR	1B_1C, 1D_1E
BGA400	1A - 1E, 2A - 2C, 3C, 4A, 4B, TR, BR	1.8, 2.5, 3.3	1A - 1E, 3C, TR, BR	3C_TR
T35				

Package	I/O Banks	Voltage (V)	DDIO Support	Merged Banks
BGA324	1A - 1E, 2A - 2C, 3A - 3C, 4A, 4B, TR, BR	1.8, 2.5, 3.3	1A - 1E, 3C, TR, BR	1B_1C, 1D_1E
BGA400	1A - 1E, 2A - 2C, 3C, 4A, 4B, TR, BR	1.8, 2.5, 3.3	1A - 1E, 3C, TR, BR	3C_TR
T55, T85, T120				
BGA324	1A - 1G, 2D - 2F, 3D, TR, BR, 4E - 4F	1.8, 2.5, 3.3	Banks 1A-1G, 3D, TR, BR	1B_1C, 1D_1E_1F_1G, 3D_TR_BR
BGA484	1A - 1G, 2A - 2F, 3D, TR, BR, 4A - 4F	1.8, 2.5, 3.3	Banks 1A-1G, 3D, TR, BR	1B_1C, 1D_1E, 1F_1G, 3D_TR_BR
BGA576	1A - 1G, 2A - 2F, 3D, TR, BR, 4A - 4F	1.8, 2.5, 3.3	Banks 1A-1G, 3D, TR, BR	1B_1C, 1D_1E_1F_1G, 3D_TR_BR

Some I/O banks are merged at the package level by sharing VCCIO pins. Merged banks have underscores (_) between banks in the name (e.g., 1B_1C means 1B and 1C are connected).

Learn more: Refer to the FPGA pinout for information on the I/O bank assignments.

Chapter 3

DDR Interface

Contents:

Some Trion FPGAs have a hardened IP interface block to communicate with off-the-shelf memories. Refer to the Package/Interface Support Matrix on page 8 to find out if your FPGA supports DDR.

About the DDR DRAM Interface

The Trion DDR PHY interface supports DDR3, DDR3L, LPDDR3, LPDDR2 memories with x16 or x32 DQ widths (depending on the FPGA) and a memory controller hard IP block. The DDR PHY supports data rates up to 1066 Mbps per lane. The memory controller provides two AXI buses to communicate with the FPGA core.

Note: The DDR PHY and controller are hard blocks; you cannot bypass the DDR DRAM memory controller to access the PHY directly for non-DDR memory controller applications.

Figure 6: (T20, T35) DDR DRAM Block Diagram

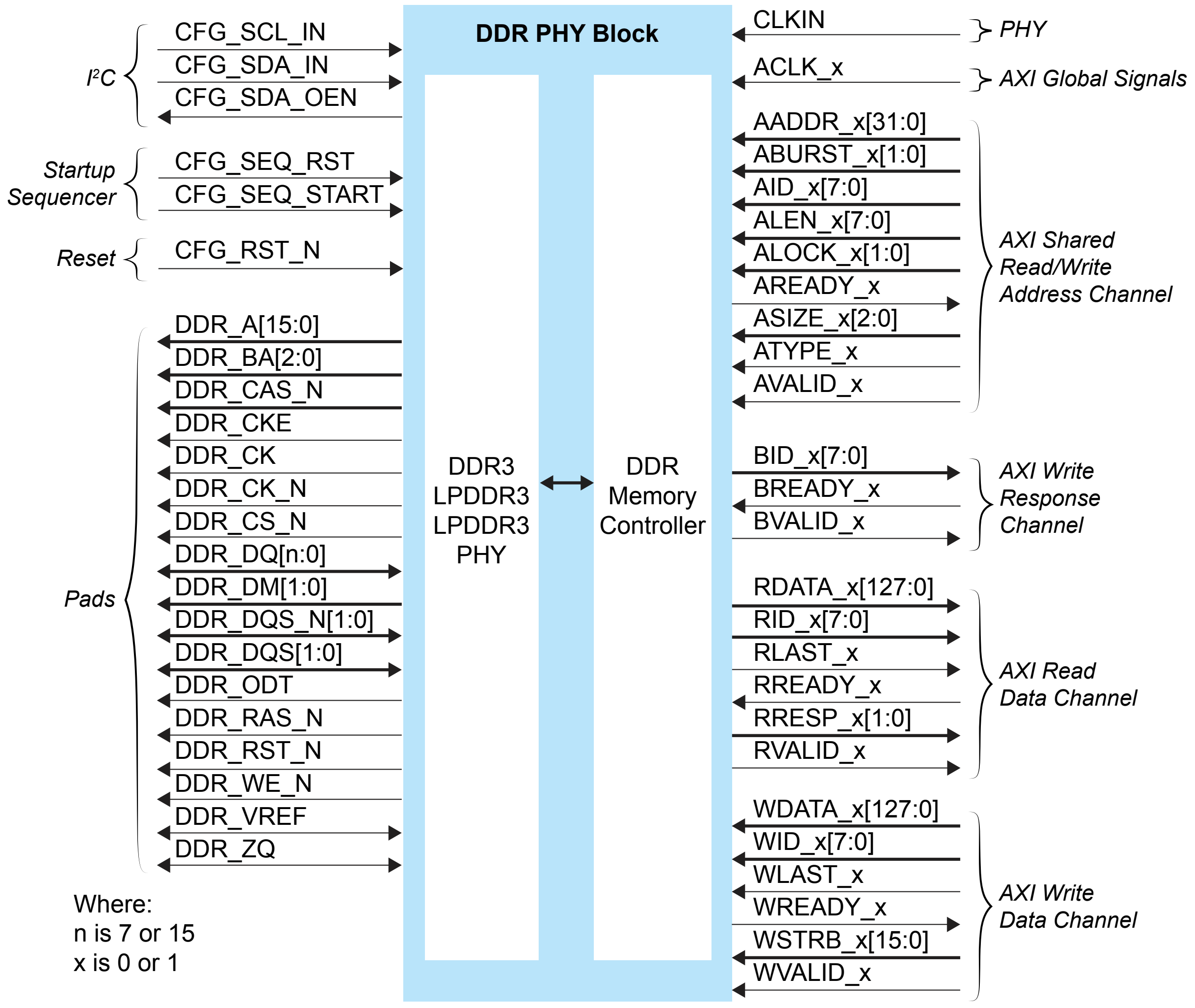
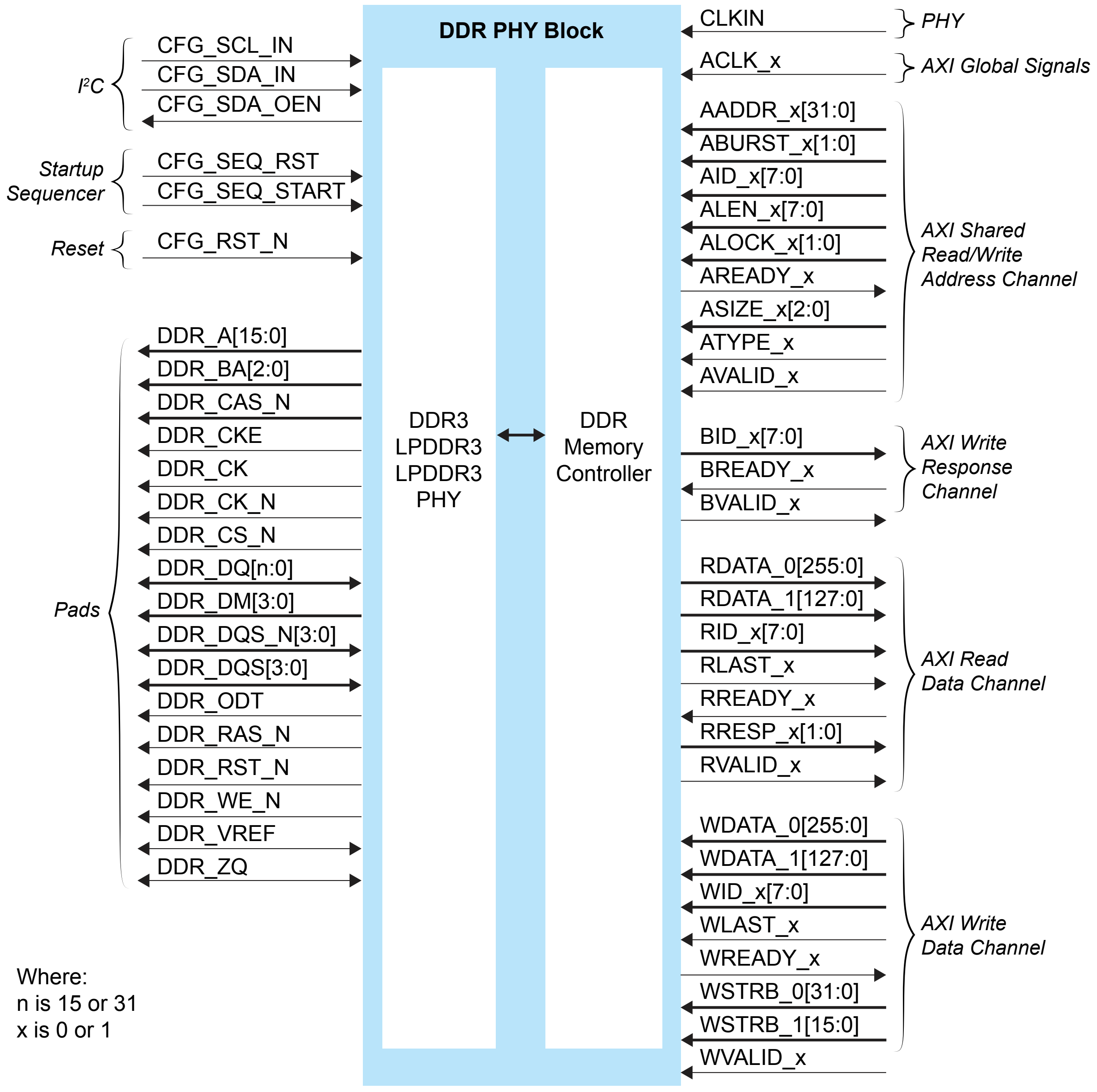
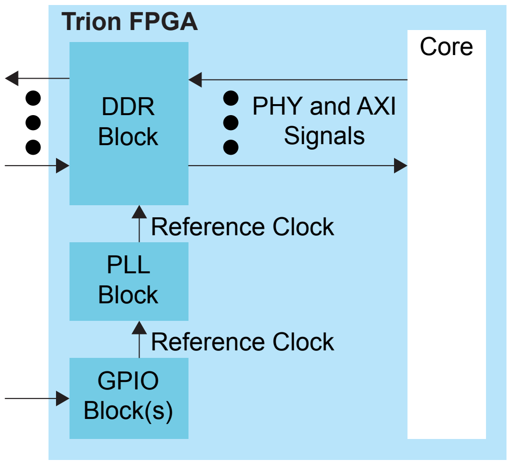


Figure 7: (T55, T85, T120) DDR DRAM Block Diagram



The DDR DRAM block supports an I^{2}C calibration bus that can read/write the DDR configuration registers. You can use this bus to fine tune the DDR PHY for high performance.

Figure 8: DDR DRAM Interface Block Diagram



Note: The PLL reference clock must be driven by I/O pads. The Efinity® software issues a warning if you do not connect the reference clock to an I/O pad. (Using the clock tree may induce additional jitter and degrade the DDR performance.) Refer to About the Advanced PLL Interface on page 75 for more information about the PLL block.

Table 5: DDR DRAM Performance

DDR DRAM Interface	Voltage (V)	Maximum Data Rate (Mbps) per Lane
DDR3	1.5	1066
DDR3L	1.35	1066
LPDDR3	1.2	1066
LPDDR2	1.2	1066

Table 6: PHY Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Description
CLKIN	Input	N/A	High-speed clock to drive the DDR PHY. A PLL must generate this clock. The clock runs at half of the PHY data rate (for example, 800 Mbps requires a 400 MHz clock). The DDR DRAM block uses the PLL_BR0 CLKOUT0 resource as the PHY clock.

Table 7: AXI Gobal Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Description
ACLK_0, ACLK_1	Input	N/A	AXI clock inputs.

Table 8: AXI Shared Read/Write Signals (Interface to FPGA Fabric)

Signal x is 0 or 1	Direction	Clock Domain	Description
AADDR_*x*[31:0]	Input	ACLK_x	Address. ATYPE defines whether it is a read or write address. It gives the address of the first transfer in a burst transaction.
ABURST_*x*[1:0]	Input	ACLK_x	Burst type. The burst type and the size determine how the address for each transfer within the burst is calculated.
AID_*x*[7:0]	Input	ACLK_x	Address ID. This signal identifies the group of address signals. Depends on ATYPE, the ID can be for a read or write address group.
ALEN_*x*[7:0]	Input	ACLK_x	Burst length. This signal indicates the number of transfers in a burst.
ALOCK_*x*[1:0]	Input	ACLK_x	Lock type. This signal provides additional information about the atomic characteristics of the transfer.
AREADY_*x*	Output	ACLK_x	Address ready. This signal indicates that the slave is ready to accept an address and associated control signals.
ASIZE_*x*[2:0]	Input	ACLK_x	Burst size. This signal indicates the size of each transfer in the burst.
ATYPE_*x*	Input	ACLK_x	This signal distinguishes whether is it is a read or write operation. 0 = read and 1 = write.
AVALID_*x*	Input	ACLK_x	Address valid. This signal indicates that the channel is signaling valid address and control information.

Table 9: AXI Write Response Channel Signals (Interface to FPGA Fabric)

Signal x is 0 or 1	Direction	Clock Domain	Description
BID_*x*[7:0]	Output	ACLK_x	Response ID tag. This signal is the ID tag of the write response.
BREADY_*x*	Input	ACLK_x	Response ready. This signal indicates that the master can accept a write response.
BVALID_*x*	Output	ACLK_x	Write response valid. This signal indicates that the channel is signaling a valid write response.

Table 10: AXI Read Data Channel Signals (Interface to FPGA Fabric)

Signal x is 0 or 1	Direction	Clock Domain	Description
RDATA_*x*[127:0]	Output	ACLK_x	(T20, T35): Read data.
RDATA_0[255:0]	Output	ACLK_0	(T55, T85, T120): AXI target 0 read data.
RDATA_1[127:0]	Output	ACLK_1	(T55, T85, T120): AXI target 1 read data.
RID_*x*[7:0]	Output	ACLK_x	Read ID tag. This signal is the identification tag for the read data group of signals generated by the slave.
RLAST_*x*	Output	ACLK_x	Read last. This signal indicates the last transfer in a read burst.
RREADY_*x*	Input	ACLK_x	Read ready. This signal indicates that the master can accept the read data and response information.
RRESP_*x*[1:0]	Output	ACLK_x	Read response. This signal indicates the status of the read transfer.
RVALID_*x*	Output	ACLK_x	Read valid. This signal indicates that the channel is signaling the required read data.

Table 11: AXI Write Data Channel Signals (Interface to FPGA Fabric)

Signal x is 0 or 1	Direction	Clock Domain	Description
WDATA_*x*[127:0]	Input	ACLK_x	(T20, T35): Write data.
WDATA_0[255:0]	Input	ACLK_0	(T55, T85, T120): AXI target 0 write data.
WDATA_1[127:0]	Input	ACLK_1	(T55, T85, T120): AXI target 1 write data.
WID_*x*[7:0]	Input	ACLK_x	Write ID tag. This signal is the ID tag of the write data transfer.
WLAST_*x*	Input	ACLK_x	Write last. This signal indicates the last transfer in a write burst.
WREADY_*x*	Output	ACLK_x	Write ready. This signal indicates that the slave can accept the write data.
WSTRB_*x*[15:0]	Input	ACLK_x	Write strobes. This signal indicates which byte lanes hold valid data. There is one write strobe bit for each eight bits of the write data bus.
WSTRB_0[31:0] WSTRB_1[15:0]	Input	ACLK_x	Write strobes. This signal indicates which byte lanes hold valid data. There is one write strobe bit for each eight bits of the write data bus.
WVALID_*x*	Input	ACLK_x	Write valid. This signal indicates that valid write data and strobes are available.

Table 12: DDR DRAM I^{2}C Interface Signals

Signal	Direction	Description
CFG_SCL_IN	Input	Clock input.
CFG_SDA_IN	Input	Data input.
CFG_SDA_OEN	Output	SDA output enable.

Table 13: DDR DRAM Startup Sequencer Signals

Signal	Direction	Description
CFG_SEQ_RST	Input	Active-high DDR configuration controller reset.
CFG_SEQ_START	Input	Start the DDR configuration controller.

Table 14: DDR DRAM Reset Signal

Signal	Direction	Description
CFG_RST_N	Input	Active-low master DDR DRAM reset. After you de-assert RST_N, you need to reconfigure and initialize before performing memory operations.

Table 15: DDR DRAM Pads

Signal	Direction	Description
DDR_A[15:0]	Output	Address signals to the memories.
DDR_BA[2:0]	Output	Bank signals to/from the memories.
DDR_CAS_N	Output	Active-low column address strobe signal to the memories.
DDR_CKE	Output	Active-high clock enable signals to the memories.
DDR_CK	Output	Active-high clock signals to/from the memories. The clock to the memories and to the memory controller must be the same clock frequency and phase.
DDR_CK_N	Output	Active-low clock signals to/from the memories.The clock to the memories and to the memory controller must be the same clock frequency and phase.
DDR_CS_N	Output	Active-low chip select signals to the memories.
DDR_DQ[*n*:0]	Bidirectional	Data bus to/from the memories. For writes, the pad drives these signals. For reads, the memory drives these signals. These signals are connected to the DQ inputs on the memories. *n* is 7, 15, or 31 depending on the FPGA and DQ width.
DDR_DM[*n*]	Output	Active-high data-mask signals to the memories. *n* is 1, 1:0, or 3:0 depending on the FPGA and DQ width.
DDR_DQS_N[*n*:0]	Bidirectional	Differential data strobes to/from the memories. For writes, the pad drives these signals. For reads, the memory drives these signals. These signals are connected to the DQS inputs on the memories. *n* is 1, 1:0, or 3:0 depending on the FPGA and DQ width.
DDR_DQS[*n*:0]	Bidirectional	
DDR_ODT	Output	ODT signal to the memories.
DDR_RAS_N	Output	Active-low row address strobe signal to the memories.
DDR_RST_N	Output	Active-low reset signals to the memories.
DDR_WE_N	Output	Active-low write enable strobe signal to the memories.
DDR_VREF	Bidirectional	Reference voltage.
DDR_ZQ	Bidirectional	ZQ calibration pin.

DDR Interface Designer Settings

The following tables describe the settings for the DDR block in the Interface Designer.

Table 16: Base Tab

Parameter	Choices	Notes
DDR Resource	None, DDR_0	Only one resource available.
Instance Name	User defined	Indicate the DDR instance name. This name is the prefix for all DDR signals.
Memory Type	DDR3, LPDDR2, LPDDR3	Choose the memory type you want to use.

Table 17: Configuration Tab

Parameter	Choices	Notes
Select Preset		The **Select Preset** button opens a list of popular DDR memory configurations. Choose a preset to populate the configuration choices. If you do not want to use a preset, you can specify the memory configuration manually.
DQ Width	x8, x16, x32	DQ bus width. The width choices vary depending on the FPGA and package.
Type	DDR3, LPDDR2, LPDDR3	Memory type.
DDR3		
Speed Grade	1066E, 1066F, 1066G, 800D, 800E	Memory speed.
Width	x8, x16	Memory width.
Density	1G, 2G, 4G, 8G	Memory density in bits.
LPDDR2		
Speed Grade	400, 533, 667, 800, 1066	Memory speed.
Width	x16, x32	Memory width. The width choices vary depending on the FPGA and package.
Density	256M, 512M, 1G, 2G, 4G	Memory density in bits.
LPDDR3		
Speed Grade	800, 1066	Memory speed.
Width	x16, x32	Memory width. The width choices vary depending on the FPGA and package.
Density	4G, 8G	Memory density in bits.

Table 18: Advanced Options Tab - FPGA Setting Subtab

Parameter	Choices	Notes
FPGA Input Termination	Varies depending on the memory type	Specify the termination value for the FPGA input/output pins.
FPGA Output Termination		

Table 19: Advanced Options Tab - Memory Mode Register Settings Subtab

Parameter	Choices	Notes
DDR3		
Burst Length	8	Specify the burst length (only 8 is supported).
DLL Precharge Power Down	On, Off	Specify whether the DLL in the memory device is off or on during precharge power-down.
Memory Auto Self-Refresh	Auto, Manual	Turn on or off auto-self refresh feature in memory device.
Memory CAS Latency (CL)	5 - 14	Specify the number of clock cycle between read command and the availability of output data at the memory device.
Memory Write CAS Latency (CWL)	5 - 12	Specify the number of clock cycle from the releasing of the internal write to the latching of the first data in at the memory device.
Memory Dynamic ODT (Rtt_WR)	Off, RZQ/2, RZQ/4	Specify the mode of dynamic ODT feature of memory device.
Memory Input Termination (Rtt_nom)	Off, RZQ/2, RZQ/4, RZQ/6, RZQ/8, RZQ/12	Specify the input termination value of the memory device.
Memory Output Termination	RZQ/6, RZQ/7	Specify the output termination value of the memory device.
Read Burst Type	Interleaved, Sequential	Specify whether accesses within a give burst are in sequential or interleaved order.
Sef-Refresh Temperature	Extended, Normal	Specify whether the self refresh temperature is normal or extended mode.
LPDDR2		
Burst Length	8	Specify the burst length (only 8 is supported).
Output Drive Strength	34.3, 40, 48, 60, 80, 120	Specify the output termination value of memory device.
Read Burst Type	Interleaved, Sequential	Specify whether accesses within a given burst are in sequential or interleaved order.

Read/Write Latency	RL=3/WL=1, RL=4/WL=2
	RL=5/WL=2, RL=6/WL=3
	RL=7/WL=4, RL=8/WL=4
		Specify the read/write latency of the memory device.

LPDDR3		
DQ ODT	Disable, RZQ1, RZQ2, RZQ4	Specify the input termination value of memory device.
Output Drive Strength	34.3 34.3 pull-down/40 pull up 34.3 pull-down/48 pull up 40 40 pull down/48 pull up 48	Specify the output termination value of memory device.

Read/Write Latency	RL=3/WL=1, RL=6/WL=3
	RL=8/WL=4, RL=9/WL=5
		Specify the read/write latency of the memory device.

Table 20: Advanced Options Tab - Memory Timing Settings Subtab

Parameter	Choices	Notes
tFAW, Four Bank Active Window (ns)	User defined	Enter the timing parameters from the memory device's data sheet.
tRAS, Active to Precharge Command Period (ns)		
tRC, Active to Actrive or REF Command Period (ns)		
tRCD, Active to Read or Write Delay (ns)		
tREFI, Average Periodic Refresh Interval (ns)		
tRFC, Refresh to Active or Refresh to Refresh Delay (ns)		
tRP, Precharge Command Period (ns)		
tRRD, Active to Active Command Period (ns)		
tRTP, Internal Read to Precharge Delay (ns)		
tWTR, Internal Write to Read Command Delay (ns)		

Table 21: Advanced Options Tab - Controller Settings Subtab

Parameter	Choices	Notes
Controller to Memory Address Mapping	BANK-ROW-COL ROW-BANK-COL ROW-COL_HIGH-BANK-COL_LOW	Specify the mapping between the address of AXI interface and column, row, and bank address of memory device.
Enable Auto Power Down	Active, Off, Pre-Charge	Specify whether to allow automatic entry into power-down mode (pre-charge or active) after a specific amount of idle time.
Enable Self Refresh Controls	No, Yes	Specify whether to enable automatic entry into self-refresh mode after specific amount of idle period.

Table 22: Advanced Options Tab - Gate Delay Tuning Settings Subtab

Parameter	Choices	Notes
Enable Gate Delay Override	On or off	Turning this option on allows you to fine-tine the gate-delay values. This is an expert only setting.
Gate Coarse Delay Tuning	0 - 5	
Gate Fine Delay Tuning	0 - 255	

Table 23: Control Tab

Option	Notes
Disable Control	When selected, this option disables calibration and user reset.
Enable Calibration	Turn on to enable optional PHY calibration pins (master reset, SCL, and SDA pins). Efinix recommends that you use the default pin names. The names are prefixed with the instance name you specified in the Base tab.
User Reset	Turn on to enable optional reset pins (master reset and sequencer start/reset). Efinix recommends that you use the default pin names. The names are prefixed with the instance name you specified in the Base tab.

Table 24: AXI 0 and AXI 1 Tabs

Parameter	Choices	Notes
Enable Target 0 Enable Target 1	On or off	Turn on to enable the AXI 0 interface. Turn on to enable the AXI 1 interface.
AXI Clock Input Pin name	User defined	Specify the name of the AXI input clock pin.
Invert AXI Clock Input	On or off	Turn on to invert the AXI clock.
Shared Read/Write Address Channel tab Write Response Channel tab Read Data Channel tab Write Data Channel tab	User defined	This tab defines the AXI signal names. Efinix recommends that you use the default names. The signals are prefixed with the instance name you specified in the Base tab.

Using the DDR Block

You can add one DDR interface block to your design (see which packages support DDR). Configuration settings are arranged in tabs.

- **Base**—Choose the resource and specify an instance name. This name becomes the prefix for all of the DDR interface signals. Also choose the **Memory Type** (DDR3, LPDDR2, or LPDDR3).
- **Configuration**—Specify the type of memory to which you want to connect. You can choose a preset configuration by clicking **Select Preset**, or you can manually specify the DQ width, speed, and density. T20 and T35 FPGAs support a x16 DQ width; T55, T85, and T120 FPGAs support x16 or X32 DQ widths.
- **Advanced Options**—Make the following settings:

Accordion Tab	Settings
FPGA Settings	Choose the input and output termination. The choices vary depending on the memory type you select in the Base tab.
Memory Mode Register Settings	Memory-specific settings. The choices vary depending on the memory type you select in the Base tab.
Memory Timing Settings	Specify the timing settings for the memory device you are using.
Controller Settings	Select how the memory address is mapped, and auto-power-down and self refresh behavior.
Gate Delay Tuning Settings	Optionally enable a gate delay override and specify coarse and fine delay tuning.

- **Control**—Optionally enable PHY calibration or soft reset. If you use this option, Efinix recommends that you keep the default pin names.
- **AXI 0**—Enable the AXI interface target 0 and specify the name of the AXI input clock pin. Efinix recommends that you keep the default names.
- **AXI 1**—Enable the AXI interface target 1 and specify the name of the AXI input clock pin. Efinix recommends that you keep the default names.

Chapter 4

GPIO Interface

Contents:

Trion® FPGAs have general-purpose I/O (GPIO) pins that allow the FPGA to communicate with other components on your circuit board. When you create your RTL design in the Efinity® software, you use the Interface Designer to add GPIO blocks for each input, output, or bi-directional pin in your design.

Trion® GPIO pins have various features, depending on the position of the pin and which package you are using. Refer to the Resource Assigner in the Interface Designer for the features of the GPIO pin you want to use.

- GPIO that provide normal functionality
- GPIO with the double-data I/O (DDIO) feature that can capture twice the data
- LVDS as GPIO where the LVDS pin acts as GPIO instead of the LVDS function

The following sections describe the GPIO interface and how to use it in your design.

About the General-Purpose I/O Logic and Buffer

The GPIO support the 3.3 V LVTTL and 1.8 V, 2.5 V, and 3.3 V LVCMOS I/O standards. The GPIOs are grouped into banks. Each bank has its own VCCIO that sets the bank voltage for the I/O standard.

Each GPIO consists of I/O logic and an I/O buffer. I/O logic connects the core logic to the I/O buffers. I/O buffers are located at the periphery of the device.

The I/O logic comprises three register types:

- *Input*—Capture interface signals from the I/O before being transferred to the core logic
- *Output*—Register signals from the core logic before being transferred to the I/O buffers
- *Output enable*—Enable and disable the I/O buffers when I/O used as output

Table 25: GPIO Modes

GPIO Mode	Description
Input	Only the input path is enabled; optionally registered. If registered, the input path uses the input clock to control the registers (positively or negatively triggered). Select the alternate input path to drive the alternate function of the GPIO. The alternate path cannot be registered. Some FPGA/package combinations support DDIO. In DDIO mode, two registers sample the data on the positive and negative edges of the input clock, creating two data streams.
Output	Only the output path is enabled; optionally registered. If registered, the output path uses the output clock to control the registers (positively or negatively triggered). The output register can be inverted. Some FPGA/package combinations support DDIO. In DDIO mode, two registers sample the data on the positive and negative edges of the input clock, creating two data streams.
Bidirectional	The input, output, and OE paths are enabled; optionally registered. If registered, the input clock controls the input register, the output clock controls the output and OE registers. All registers can be positively or negatively triggered. Additionally, the input and output paths can be registered independently. The output register can be inverted.
Clock output	Clock output path is enabled.

During configuration, all GPIO pins are tristated and configured in weak pull-up mode.

By default, unused GPIO pins are tristated and configured in weak pull-up mode. You can change the default mode to weak pull-down in the Interface Designer.

Table 26: Features for GPIO and LVDS as GPIO by FPGA and Package

Package	Supported Features	
	GPIO	LVDS GPIO
T4/T8		
BGA49 BGA81	Schmitt Trigger Variable Drive Strength Pull-up Pull-down Slew Rate	–
T8/T13/T20		
WLCSP80 QFP144 BGA169 BGA256	DDIO Schmitt Trigger Variable Drive Strength Pull-up Pull-down Slew Rate	Pull-up
T20/T35/T55/T85/T120		
BGA324 BGA400 BGA484 BGA576	DDIO Schmitt Trigger Variable Drive Strength Pull-up Pull-down Slew Rate	Variable Drive Strength Pull-up Slew Rate

Simple I/O Buffer

T4/T8 FPGAs in BGA49 and BGA81 packages have simple I/O interface with logic and a buffer.

Figure 9: T8/T4 I/O Interface Block

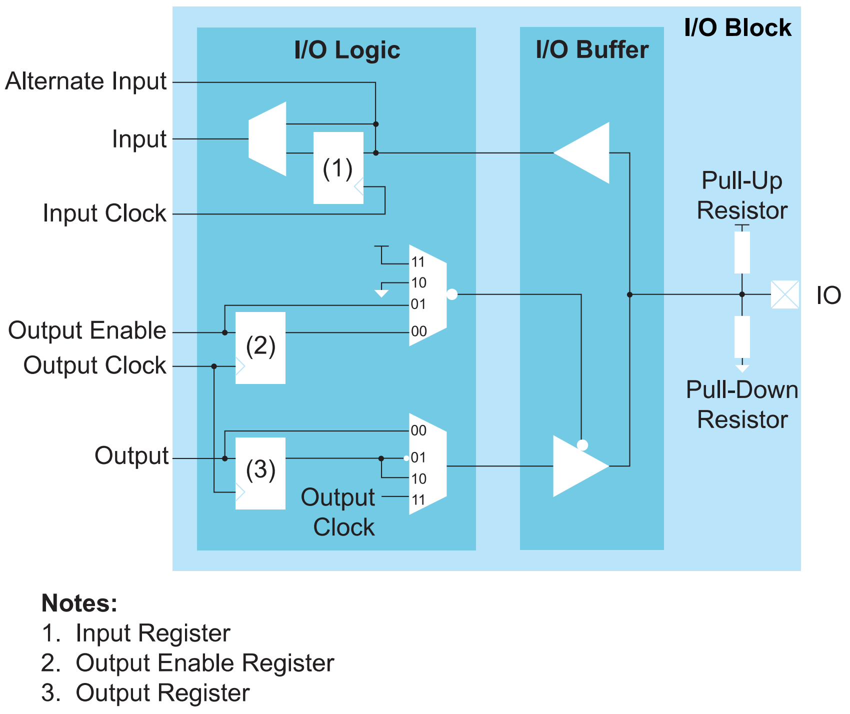


Table 27: GPIO Signals

Signal	Direction	Description
IN	Output	Input data from the GPIO pad to the core fabric.
ALT	Output	Alternative input connection (in the Interface Designer, the input **Register Option** is **none**). Alternative connections are GCLK, GCTRL, and PLL_CLKIN.
OUT	Input	Output data to GPIO pad from the core fabric.
OE	Input	Output enable from core fabric to the I/O block. Can be registered.
OUTCLK	Input	Core clock that controls the output and OE register. This clock is not visible in the user netlist.
INCLK	Input	Core clock that controls the input register. This clock is not visible in the user netlist.

Table 28: GPIO Pads

Signal	Direction	Description
IO	Bidirectional	GPIO pad.

Complex I/O Buffer

T8 (QFP144 only), T13, T20, T35, T55, T85, and T120 FPGAs have a complex I/O interface with logic and a buffer.

Table 29: GPIO Signals (Interface to FPGA Fabric)

Signal	Direction	Description
IN[1:0]	Output	Input data from the GPIO pad to the core fabric. IN0 is the normal input to the core. In DDIO mode, IN0 is the data captured on the positive clock edge (HI pin name in the Interface Designer) and IN1 is the data captured on the negative clock edge (LO pin name in the Interface Designer).
ALT	Output	Alternative input connection (in the Interface Designer, **Register Option** is **none**). Alternative connections are GCLK, GCTRL, PLL_CLKIN MIPI_CLKIN and PLL_EXTFB.
OUT[1:0]	Input	Output data to GPIO pad from the core fabric. OUT0 is the normal output from the core. In DDIO mode, OUT0 is the data captured on the positive clock edge (HI pin name in the Interface Designer) and OUT1 is the data captured on the negative clock edge (LO pin name in the Interface Designer).
OE	Input	Output enable from core fabric to the I/O block. Can be registered.
OUTCLK	Input	Core clock that controls the output and OE registers. This clock is not visible in the user netlist.
INCLK	Input	Core clock that controls the input registers. This clock is not visible in the user netlist.

Table 30: GPIO Pads

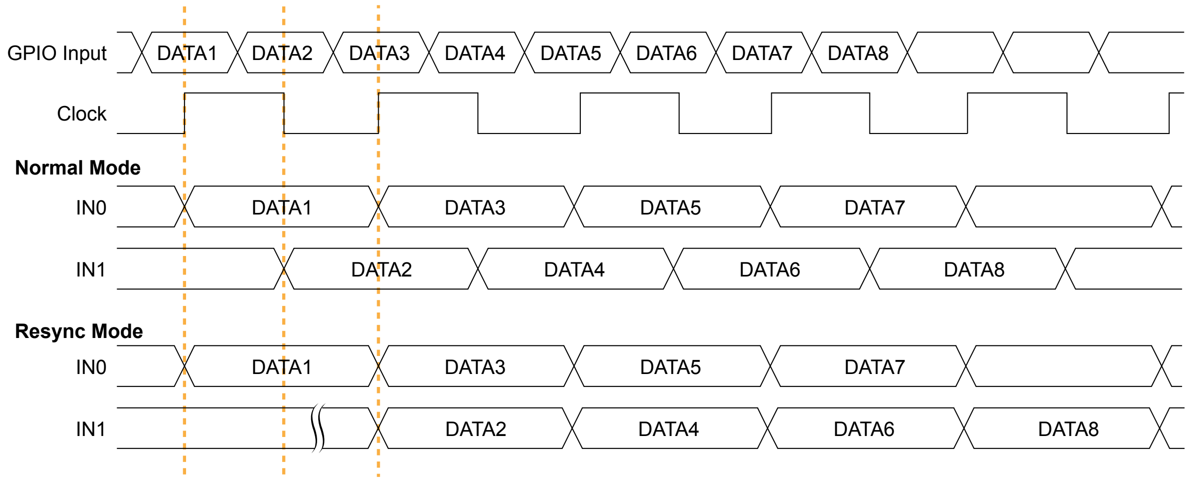
Signal	Direction	Description
IO	Bidirectional	GPIO pad.

Double-Data I/O

Some Trion FPGAs support double data I/O (DDIO) on input and output registers. In this mode, the DDIO register captures data on both positive and negative clock edges. The core receives 2 bit wide data from the interface.

In normal mode, the interface receives or sends data directly to or from the core on the positive and negative clock edges. In resync mode, the interface resynchronizes the data to pass both signals on the positive clock edge only.

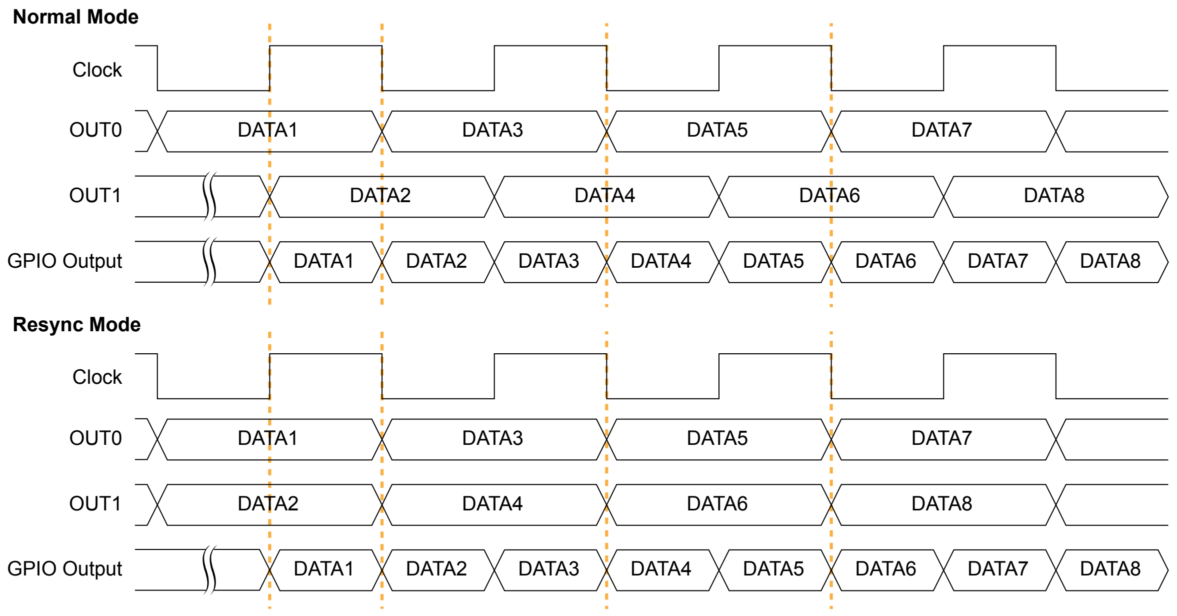
Figure 10: DDIO Input Timing Waveform



In resync mode, the IN1 data captured on the falling clock edge is delayed one half clock cycle.

In the Interface Designer, IN0 is the HI pin name and IN1 is the LO pin name.

Figure 11: DDIO Output Timing Waveform



In the Interface Designer, OUT0 is the HI pin name and OUT1 is the LO pin name.

Using the GPIO Block

This block defines the functionality of the general-purpose I/O (GPIO) pins. The mode you select determines the GPIO capabilities and which settings you can configure. GPIO modes are: input, output, inout, clkout, and none.

You can assign GPIO to dedicated GPIO resources or to LVDS resources. When you use LVDS resources as GPIO, some features are unavailable, depending on the FPGA. When you check the interface design, the software compares your selections to the resource you assigned to the GPIO block. If the resource does not support your selection(s), the software reports it.

Create a GPIO

To create a new GPIO block, select GPIO in the Design Explorer and then click the Create Block button.

1. Specify the instance name.
2. Choose the Mode (input, output, inout, clkout, or none).
3. Set the options as described in the following sections.
4. Assign a resource for the signal using the Resource Assigner.

Note: You can set the default state of unused GPIO. Click the **GPIO(*n*)** category under Design Explorer. In the Block Editor to the right, select the unused state (**input with weak pull up** or **input with weak pull down**).

Note: When using LVDS pins as GPIO, make sure to leave at least 2 pairs of unassigned LVDS pins between any GPIO and LVDS pins in the same bank. This separation reduces noise. The Efinity software issues an error if you do not leave this separation.

Input Mode

Use input mode for input signals.

Table 31: Input Mode Options

Option	Choices	Description
Connection Type	normal, gclk, gctrl, pll_clkin, pll_extfb, mipi_clkin	Some pins have alternate functions, and you use this option to choose the function. (This option only applies to pins that have alternate functions. Refer to the data sheet for your FPGA for pin information.) For example, a PLL can use a GPIO with an alternate connection type as a reference clock. **Note:** If you set the connection type to **pll_clkin**, **pll_extfb**, or **mipi_clkin**, the signal is also available as a regular input to the core.
Register Option	register, none	Choose whether the input is registered. For FPGAs that have DDIO, if you choose **register**: • Define an input clock pin name. • Turn clock inversion on or off. • Under **Double Data I/O Option**, select one of the following: **none** — Do not use double data I/O. **normal** — Data is passed to the core on both the positive and negative clock edges **resync** — Data is resynchronized to pass both data signals on the positive clock edge. `<pin name>_hi` is the positive edge and `<pin name>_lo` is the negative edge.
Pull Option	none, weak pullup, weak pulldown	Specify if you want a pull option.
Enable Schmitt Trigger	On or off	Optionally enable a Schmitt trigger.

Output Mode

Use output mode for output signals.

Table 32: Output Mode Options

Option	Choices	Description
Constant Output	none, 1, 0	Choose whether the output is VCC (**1**) or GND (**0**). Otherwise, leave this option as none.

Option	Choices	Description
Register Option	none, register, inv_register	Choose whether the output is registered or has an inverted register. For FPGAs that have DDIO, if you choose **register**: • Define an output clock pin name. • Turn clock inversion on or off. • Under **Double Data I/O Option**, select one of the following: **none**: Do not use double data I/O. **normal**: Data is passed to the core on both the positive and negative clock edges **resync**: Data is resynchronized to pass both data signals on the positive clock edge. `<pin name>_hi` is the positive edge and `<pin name>_lo` is the negative edge. The invert register option (**inv_register**) does not support DDIO.
Drive Strength	1, 2, 3, 4	Choose the drive strength level.
Enable Fast Slew Rate	On or off	Optionally enable slew rate.

Inout Mode

Use **inout** mode for bidirectional signals. Inout mode has the same options for the input and output as the input and output modes.

Inout mode also has an output enable signal (optionally registered) to enable or disable the output buffer. The pin name you specify should be the same as the one you use in your RTL design. Setting the output enable signal to high ("1") in your RTL design enables the output buffer.

Learn more: For information on how to create a tri-state buffer, refer to "How do I create a Tri-State Buffer" in the Support Center Knowledgebase.

Clock Output Mode

Use **clkout** mode for clock output signals. You do not need to name the pin, but you do need to specify the output clock **Pin Name**.

None

Use **none** for unused signals. Specify whether the unused signal should have a weak pullup (default) or pulldown.

Using LVDS as GPIO

You can use LVDS as GPIO by simply creating a GPIO block and assigning an LVDS resoure to it. When you use LVDS resources as GPIO, some features are unavailable, depending on the FPGA as described in Table 26: Features for GPIO and LVDS as GPIO by FPGA and Package on page 31.

Assign a resource for the signal using the Resource Assigner.

Important: In the same bank, when you are using an LVDS pin as a GPIO, do not assign it to a pin that is closer than 2 pairs away from any LVDS pins you are using in LVDS mode. That is, leave 2 unused pairs between any LVDS GPIO pins and LVDS pins, otherwise the pins will interfere with each other.

Using the GPIO Bus Block

The GPIO bus block is an easy way to add a group of GPIO blocks and make settings for the signal group.

1. Click **Create New Bus**. The Add New Bus wizard opens.
2. Specify a bus name, the width, and the mode (input, output, or inout) and click **Next**.
3. The wizard displays options for input, output, or inout, depending on the mode you selected. Refer to Using the GPIO Block on page 35 for a description of these options. Make your selections and click **Next**.
4. Review the bus properties and click **Finish**. The software adds the new bus under GPIO.

After you create a bus, you can make additional settings for each signal.

1. Expand **GPIO > <bus name>**.
2. Make any block-specific settings in the Block Editor.
3. Assign a resource for the signal using the Resource Assigner.
4. Save.

Note: Any changes that you make to individual bus members are over written if you later edit the bus.

Chapter 5

JTAG User TAP Interface

Contents:

Trion® FPGAs have dedicated JTAG pins to support configuration and boundary scan testing.

Boundary-Scan and JTAG Mode

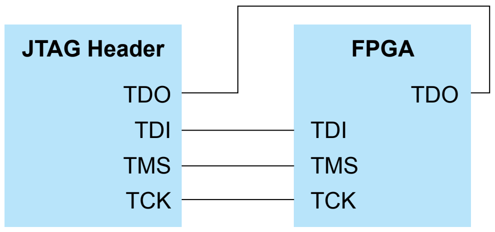
The JTAG serial configuration mode is popular for prototyping and board testing. The four-pin JTAG boundary-scan interface is commonly available on board testers and debugging hardware. Efinix FPGAs support IEEE standard 1149.1 - 2001.

Learn more: Refer to the following web sites for more information about the JTAG interface:
http://ieeexplore.ieee.org/document/6515989/
https://en.wikipedia.org/wiki/JTAG
Refer to AN 021: Performing Boundary-Scan Testing in Trion FPGAs for information about boundary-scan testing.

Table 33: Supported JTAG Boundary-Scan Instructions

Instruction	Binary Code [3:0]	Description
SAMPLE/PRELOAD	0010	Enables the boundary-scan SAMPLE/PRELOAD operation
EXTEST	0000	Enables the boundary-scan EXTEST operation
BYPASS	1111	Enables BYPASS
IDCODE	0011	Enables shifting out the IDCODE
PROGRAM	0100	JTAG configuration
ENTERUSER	0111	Changes the FPGA into user mode.

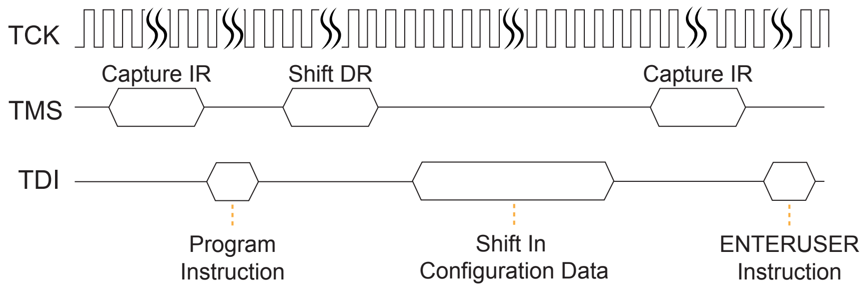
Figure 12: JTAG Programming (One FPGA)



When configuration ends, the JTAG host issues the ENTERUSER instruction to the FPGA. After `CDONE` goes high and the FPGA receives the ENTERUSER instruction, the FPGA waits for t_{USER} to elapse, and then it goes into user mode.

Note: The FPGA may go into user mode before t_{USER} has elapsed. Therefore, you should keep the system interface with the FPGA in reset until t_{USER} has elapsed.

Figure 13: JTAG Programming Waveform



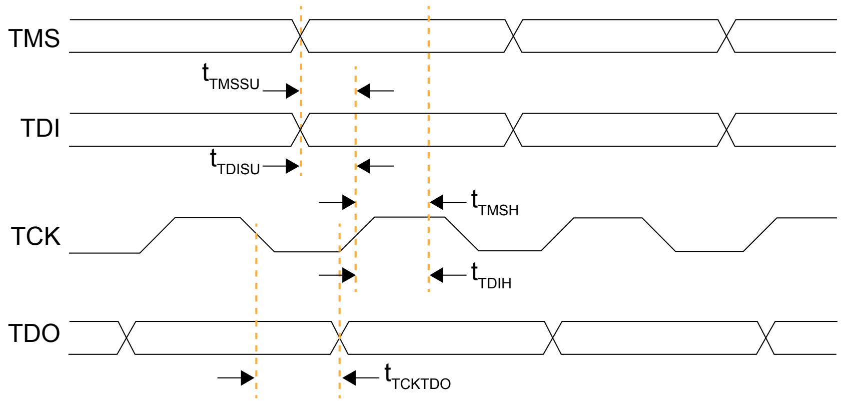
Design Considerations

- Because the `TCK` and `TMS` signals connect devices in the JTAG chain, they must have good signal quality.
- `TCK` should transition monotonically at the receiving devices and should be terminated correctly. Poor `TCK` quality can limit the maximum frequency you can use for configuration.
- Buffer `TMS` and `TCK` so they have sufficient drive strength at all receiving devices.
- Ensure that the logic high voltage is compatible with all devices in the JTAG chain.
- If your chain contains devices from different vendors, you might need to drive optional JTAG signals, such as `TRST` and enables.

Note: For JTAG programming, T4, T8, T13, and T20 FPGAs use the `CRESET_N` and `SS_N` pins in addition to the standard JTAG pins. Refer to JTAG Programming with FTDI Chip Hardware for more details. You can use the standard 4 JTAG pins and any JTAG cable for programming with SPI Active using JTAG mode or for other JTAG functions.

Timing Parameters

Figure 14: Boundary-Scan Timing Waveform



Learn more: Refer to the FPGA data sheet for timing specifications.

Refer to the Virtual I/O Debug Core section in the Efinity Software User Guide for more information about JTAG User TAP interface.

Using the JTAG User TAP Block

Add the JTAG User TAP block to your interface if you want to use the FPGA JTAG pins to communicate with the design running in the core.

You specify the instruction to use with the **JTAG Resource** setting. Trion FPGAs have two JTAG User TAP blocks. To use both USER1 and USER2, add 2 blocks to your interface design, one for each resource.

Chapter 6

LVDS Interface

Contents:

Some Trion® FPGAs support low-voltage differential signaling (LVDS) on their pins. LVDS offers the advantage of running at high speeds with low power. Refer to the Package/ Interface Support Matrix on page 8 to find out if your FPGA supports LVDS. The following sections describe the LVDS pins and how to use them in your Efinity® RTL design.

About the LVDS Interface

The LVDS hard IP transmitters and receivers operate independently.
- LVDS TX consists of LVDS transmitter and serializer logic.
- LVDS RX consists of LVDS receiver, on-die termination, and de-serializer logic.

Trion® FPGAs have one or more PLLs for use with the LVDS receiver, depending on which FPGA you use.

You can use the LVDS TX and LVDS RX channels as single-ended GPIO pins, see Table 2. The voltage supported depends on the FPGA.

Note: When using LVDS as GPIO, make sure to leave at least 2 pairs of unassigned LVDS pins between any GPIO and LVDS pins in the same bank. This separation reduces noise. The Efinity software issues an error if you do not leave this separation.

The LVDS hard IP has these features:

- Dedicated LVDS TX and RX channels (the number of channels depends on the FPGA and package)
- Up to 600 or 800 Mbps for LVDS data transmit or receive (depending on the FPGA and package)
- Supports serialization and deserialization factors: 8:1, 7:1, 6:1, 5:1, 4:1, 3:1, and 2:1
- Ability to disable serialization and deserialization
- Source synchronous clock output edge-aligned with data for LVDS transmitter and receiver
- 100 Ω on-die termination resistor for the LVDS receiver

LVDS TX

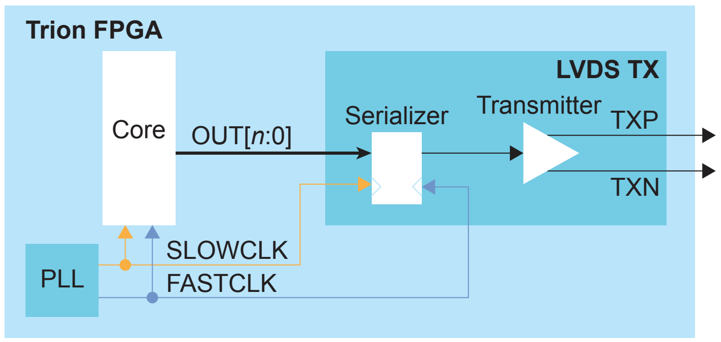


Figure 15: LVDS TX Interface Block Diagram

Table 34: LVDS TX Signals (Interface to FPGA Fabric)

Signal	Direction	Notes
OUT[*n*-1:0]	Input	Parallel output data where *n* is the serialization factor. A width of 1 bypasses the serializer.
FASTCLK	Input	Fast clock to serialize the data to the LVDS pads.
SLOWCLK	Input	Slow clock to latch the incoming data from the core.

Table 35: LVDS TX Pads

Pad	Direction	Description
TXP	Output	Differential P pad.
TXN	Output	Differential N pad.

The following waveform shows the relationship between the fast clock, slow clock, TX data going to the pad, and byte-aligned data from the core.

Figure 16: LVDS Timing Example Serialization Width of 8

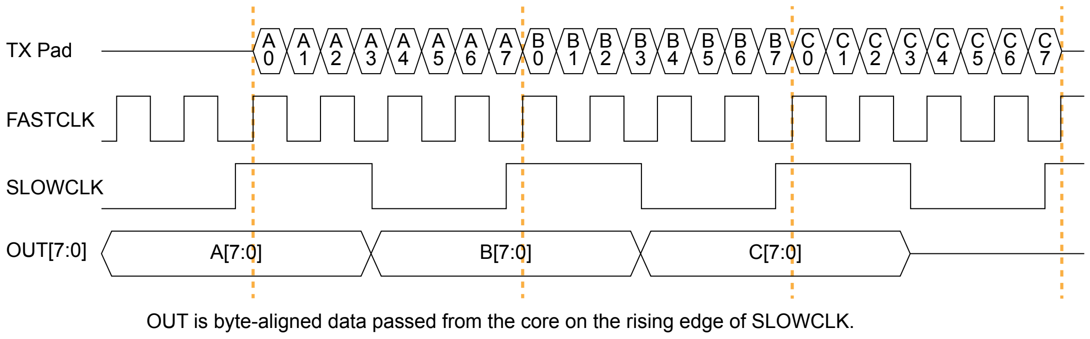

TX Pad
A0 A1 A2 A3 A4 A5 A6 A7 B0 B1 B2 B3 B4 B5 B6 B7 C0 C1 C2 C3 C4 C5 C6 C7
FASTCLK
SLOWCLK
OUT[7:0]
A[7:0]
B[7:0]
C[7:0]
OUT is byte-aligned data passed from the core on the rising edge of SLOWCLK.

Figure 17: LVDS Timing Data and Clock Relationship Width of 8 (Parallel Clock Division=1)

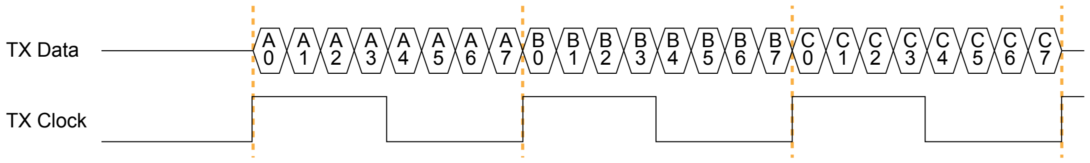

TX Data
A0 A1 A2 A3 A4 A5 A6 A7 B0 B1 B2 B3 B4 B5 B6 B7 C0 C1 C2 C3 C4 C5 C6 C7
TX Clock

Figure 18: LVDS Timing Data and Clock Relationship Width of 7 (Parallel Clock Division=1)

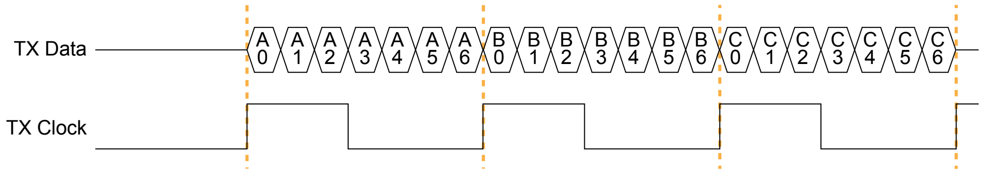

TX Data
A0 A1 A2 A3 A4 A5 A6 B0 B1 B2 B3 B4 B5 B6 C0 C1 C2 C3 C4 C5 C6
TX Clock

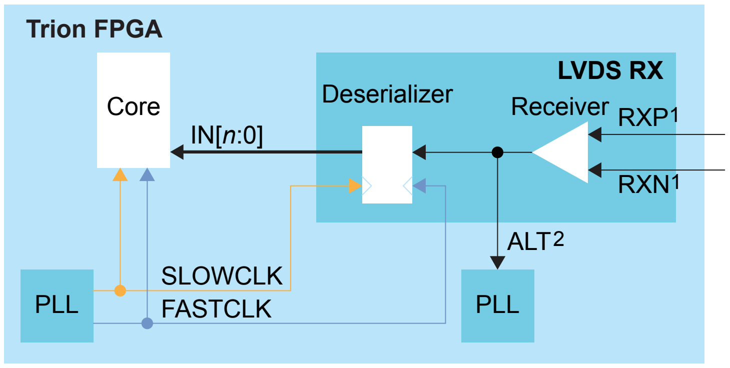
Table 36: LVDS TX Settings in Efinity® Interface Designer

Parameters	Choices	Notes
Mode	serial data output or reference clock output	**serial data output**–Simple output buffer or serialized output. **reference clock output**–Use the transmitter as a clock output. When choosing this mode, the **Serialization Width** you choose should match the serialization for the rest of the LVDS bus.
Parallel Clock Division	1, 2	**1**–The output clock from the LVDS TX lane is parallel clock frequency. **2**–The output clock from the TX lane is half of the parallel clock frequency.
Enable Serialization	On or off	When off, the serializer is bypassed and the LVDS buffer is used as a normal output.
Serialization Width	2, 3, 4, 5, 6, 7, or 8	Supports 8:1, 7:1, 6:1, 5:1, 4:1, 3:1, and 2:1.
Reduce VOD Swing	On or off	When true, enables reduced output swing (similar to slow slew rate).
Output Load	Varies by FPGA.	Output load in pF. [5][6]

[5] For T20BGA324, T20BGA400, T35, T55, T85 and T120 FPGAs, use an output load of 7 pF or higher to achieve the maximum throughput of 800 Mbps.
[6] For T8Q144 FPGAs, use an output load of 7 pF or higher to achieve the maximum throughput of 600 Mbps.

LVDS RX

Figure 19: LVDS RX Interface Block Diagram



1. There is a ~30k Ω internal weak pull-up to VCCIO (3.3V).
2. Only available for an LVDS RX resource in bypass mode (deserialization width is 1).

Table 37: LVDS RX Signals (Interface to FPGA Fabric)

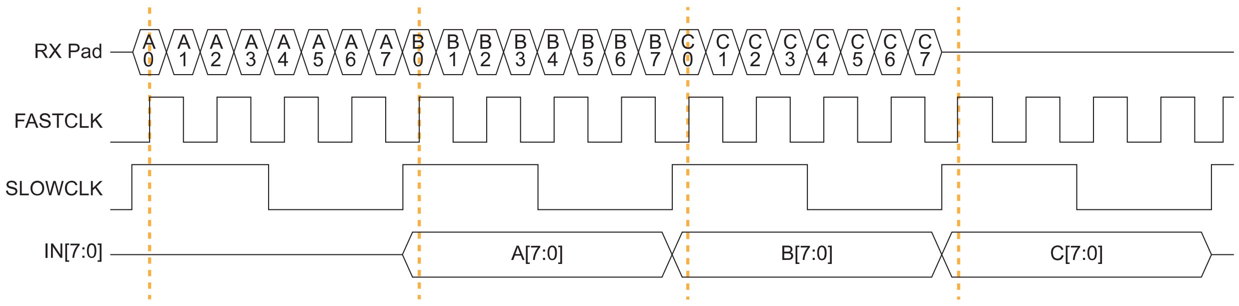
Signal	Direction	Notes
IN[*n*-1:0]	Output	Parallel input data where *n* is the de-serialization factor. A width of 1 bypasses the deserializer.
ALT	Output	Alternative input, only available for an LVDS RX resource in bypass mode (deserialization width is 1; alternate connection type). Alternative connections are PLL_CLKIN and PLL_EXTFB.
FASTCLK	Input	Fast clock to de-serialize the data from the LVDS pads.
SLOWCLK	Input	Slow clock to latch the incoming data to the core.

Table 38: LVDS RX Pads

Pad	Direction	Description
RXP	Input	Differential P pad.
RXN	Input	Differential N pad.

The following waveform shows the relationship between the fast clock, slow clock, RX data coming in from the pad, and byte-aligned data to the core.

Figure 20: LVDS RX Timing Example Serialization Width of 8


RX Pad
A0 A1 A2 A3 A4 A5 A6 A7 B0 B1 B2 B3 B4 B5 B6 B7 C0 C1 C2 C3 C4 C5 C6 C7
FASTCLK
SLOWCLK
IN[7:0]
A[7:0]
B[7:0]
C[7:0]

IN is byte-aligned data passed to the core on the rising edge of SLOWCLK.

Table 39: LVDS RX Settings in Efinity® Interface Designer

Parameter	Choices	Notes
Connection Type	normal, pll_clkin, pll_extfb	**normal**–Regular RX function. **pll_clkin**–Use the PLL CLKIN alternate function of the LVDS RX resource. **pll_extfb**–Use the PLL external feedback alternate function of the LVDS RX resource.
Enable Deserialization	On or off	When off, the de-serializer is bypassed and the LVDS buffer is used as a normal input.
Deserialization Width	2, 3, 4, 5, 6, 7, or 8	Supports 8:1, 7:1, 6:1, 5:1, 4:1, 3:1, and 2:1.
Enable On-Die Termination	On or off	When on, enables an on-die 100-ohm resistor.

Using the LVDS Block

The LVDS block defines the functionality of the LVDS pins. You can choose whether the block is a transmitter (TX) or receiver (RX).

LVDS TX

Table 40: LVDS TX Options

Option	Choices	Description
Instance Name	User defined	
LVDS Resource	Resource list	Choose a resource.
Mode	serial data output	Use the transmitter as a simple output buffer or serialized output.
	reference clock output	Use the transmitter as a clock output. Choose 1 or 2 for the parallel clock division. Specify the serial and parallel clocks. When choosing this mode, the serialization width should match the serialization for the rest of the LVDS bus.
Output Pin/Bus Name	User defined	Output pin or bus that feeds the LVDS transmitter parallel data. The width should match the serialization factor.
Output Enable Pin Name	User defined	Use with serial data output mode.
Enable Serialization	On	Use as a simple buffer.
	Off	Use as an LVDS serializer (half-rate mode): • Choose a value of 2 - 8. • Specify the serial clock and parallel clock.
Output Load	3, 5, 7, 10	For T8Q144 FPGAs, use an output load of 7 pF or higher to achieve the maximum throughput of 600 Mbps.
Reduce VOD Swing	On or off	Turn on to reduce the differential voltage swing.

The maximum LVDS rate is 800 Mbps. The serial clock frequency = parallel clock frequency * (serialization / 2). The serial clock must use the 90 degree phase shift and both clocks must come from the same PLL.

Note: Efinix® recommends that you select a PLL post divider of 2 or higher and an output divider of 2. These settings provide a more stable clock signal for faster speeds.

The serial clock (also known as the fast clock) outputs data to the pin, the parallel clock (also known as the slow clock) transfers it from the core. An equation defines the relationship between the clocks. For LVDS TX the parallel clock captures data from the core and the serial clock outputs it to the LVDS buffer.

New data is output on both edges of the serial clock.

LVDS RX

Table 41: LVDS RX Options

Option	Choices	Description
Instance Name	User defined	
LVDS Resource	Resource list	Choose a resource.
Connection Type	normal	LVDS RX function.
	pll_clkin	Alternate function. Use as PLL reference clock.
	pll_extfb	Alternate function. Use as PLL external feedback.
Input Pin/Bus Name	User defined	Input pin or bus that feeds the LVDS transmitter parallel data. The width should match the deserialization factor.
Dynamic Enable Pin Name	User defined	Dynamically enables or disables the LVDS RX buffer. Disabling the buffer can reduce power consumption when the pin is not in use.
On-Die LVDS Termination	On or off	Turn on to enable on-die termination.
Enable Deserialization	Off	Use as a simple buffer.
	On	Use as an LVDS deserializer: • Choose a width of 2 - 8. • Specify the serial clock and parallel clock.

The serial clock (also known as the fast clock) captures data from the pin, the parallel clock (also known as the slow clock) transfers it to the core. An equation defines the relationship between the clocks.

The maximum LVDS rate is 800 Mbps. The serial clock frequency = parallel clock frequency * (serialization / 2). The serial clock must use the 90 degree phase shift and both clocks must come from the same PLL.

Note: Efinix® recommends that you select a PLL post divider of 2 or higher and an output divider of 2. These settings provide a more stable clock signal for faster speeds. If the LVDS receiver speed is 600 Mbps or higher, the Efinity® software issues a warning if you select 1 as the PLL post divider value.

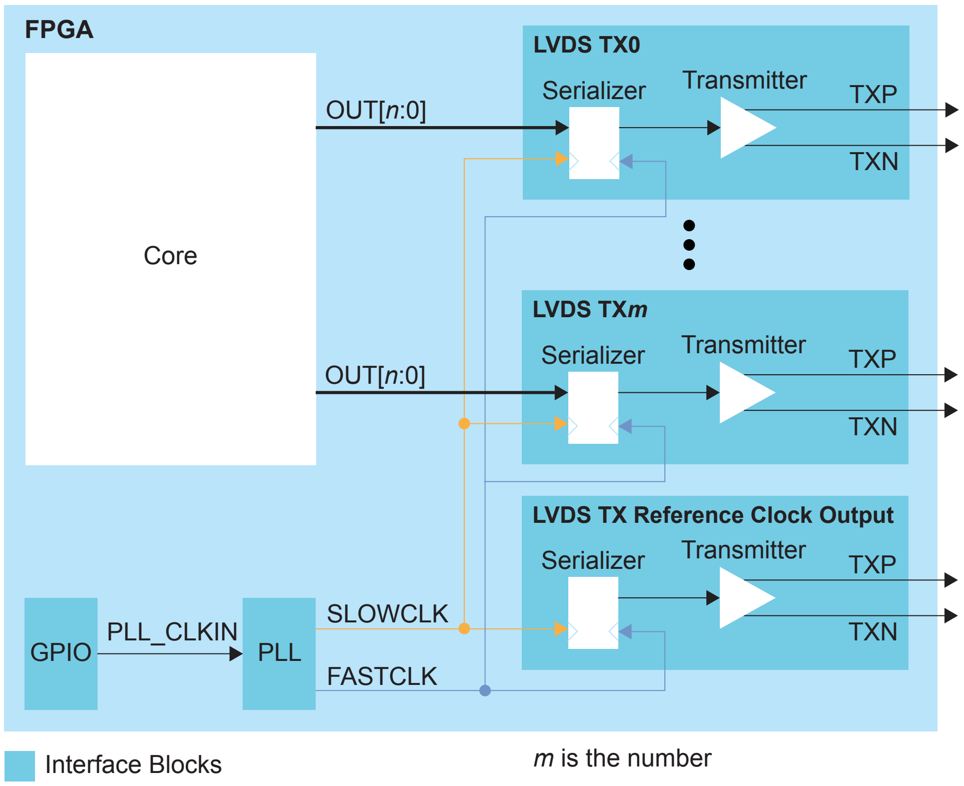
Create an LVDS TX or RX Interface

You build a complete interface using the Efinity® Interface Designer and LVDS, PLL, and GPIO blocks.

Create an LVDS TX Interface

The following figure shows a completed LVDS TX interface, where *n* is the serialization width and *m* is the number of TX lanes.

Figure 21: Complete LVDS TX Interface Block Diagram



Follow these steps to build an LVDS TX interface using the Efinity® Interface Designer.

1. Add a PLL block with the following settings:

Option	Description
Resource	You can use any PLL resource. T13/T20 BGA169 and BGA256 only: if you also want to create an RX interface, do not select PLL_BR0 because it is the only PLL the RX interface can use.
Reference Clock Mode	External
Reference Clock Frequency	Any
Output Clock	For LVDS serializer widths 2 - 8, define the output clocks so that you have one for the fast clock (serial) and one for the slow clock (parallel).

Option	Description
	Set the relationship between the clocks such that the serial clock frequency = parallel clock frequency * (serialization / 2). The serial clock must use the 90 degree phase shift.

2. Add a GPIO block with these settings to provide the reference clock input to the PLL:

Option	Description
Mode	Input
Pin Name	Any
Connection Type	pll_clkin
GPIO Resource	Assign the dedicated PLL_CLKIN pin that corresponds to the PLL you chose.

3. Add an LVDS TX block with these settings:

Option	Description
LVDS Type	Transmitter (TX)
LVDS Resource	Any channel
Mode	Serial data output
Enable Serialization	On
Serialization Width	*n*
Output Pin/ Bus Name	Any
Serial Clock Pin Name	Use the fast clock output name that corresponds to the PLL you chose.
Parallel Clock Pin Name	Use the slow clock output name that corresponds to the PLL you chose.

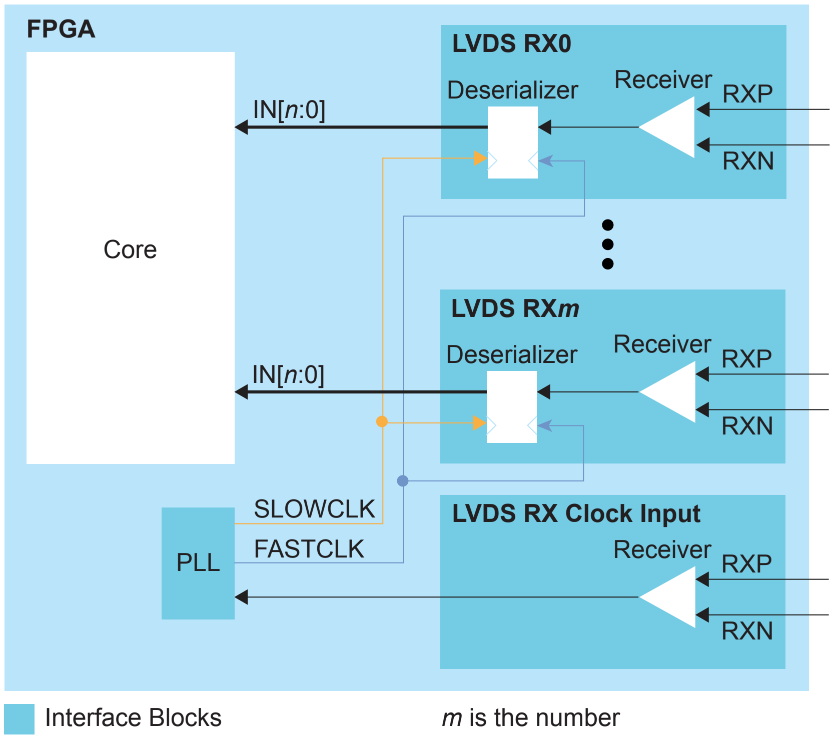
4. Repeat step 3 for each LVDS TX data lane you want to implement.
5. Add another LVDS block that will serve as the LVDS TX reference clock output:

Option	Description
LVDS type	Transmitter (TX)
LVDS resource	Any channel
Mode	Reference clock output
Enable Serialization	On
Serialization width	*n*
Output pin/ bus name	Any
Parallel clock division	1: The output clock from the LVDS TX lane is parallel clock frequency. 2: The output clock from the TX lane is half of the parallel clock frequency.
Serial clock pin name	Specify the fast clock output name that corresponds to the PLL you chose.
Parallel clock pin name	Use the slow clock output name that corresponds to the PLL you chose.

Create an LVDS RX Interface

The following figure shows a completed LVDS RX interface, where *n* is the deserialization width and *m* is the number of RX lanes.

Figure 22: Complete LVDS RX Interface Block Diagram



Follow these steps to build an LVDS RX interface using the Efinity® Interface Designer.

1. Add an LVDS RX block to act as the PLL reference clock input:

Option	Description
LVDS Type	Receiver (RX)
LVDS Resource	T13/T20 BGA169 and BGA256 only: GPIOB_CLK0.lvds
Connection Type	pll_clkin
Input Pin/Bus Name	Use the clock LVDS RX clock output name as the incoming clock.

2. Add a PLL block with the following settings:

Option	Description
Resource	T13/T20 BGA169 and BGA256 only: Select BR_PLL0, which is the only PLL the LVDS RX interface can use.
Reference Clock Mode	External
Reference Clock Frequency	Set the reference clock frequency to match the clock coming from the LVDS RX reference clock you created in step 1.
Output Clock	For LVDS deserializer widths 2 - 8, define the output clocks so that you have one for the fast clock (serial) and one for the slow clock (parallel). Set the relationship between the clocks such that the serial clock frequency = parallel clock frequency * (serialization / 2). The serial clock must use the 90 degree phase shift.

3. Add an LVDS RX block with these settings:

Option	Description
LVDS Type	Receiver (RX)
LVDS Resource	Any channel
Enable Deserialization	On
Deserialization Width	*n*
Output Pin/ Bus Name	Any
Serial Clock Pin Name	Use the fast clock output name that corresponds to the PLL you chose.
Parallel Clock Pin Name	Use the slow clock output name that corresponds to the PLL you chose.

4. Repeat step 3 for each LVDS RX data lane you want to implement.

Chapter 7

MIPI CSI-2 Interface

Contents:

Some Trion FPGAs have a hardened Mobile Industry Processor Interface (MIPI) block to communicate with cameras and sensors. Refer to the Package/Interface Support Matrix on page 8 to find out if your FPGA supports MIPI.

About the MIPI Interface

The MIPI CSI-2 interface is the most widely used camera interface for mobile.[7]. You can use this interface to build single- or multi-camera designs for a variety of applications.

Trion FPGAs can include hardened MIPI D-PHY blocks (4 data lanes and 1 clock lane) with MIPI CSI-2 IP blocks. The MIPI RX and MIPI TX can operate independently with dedicated I/O banks.

Note: The MIPI D-PHY and CSI-2 controller are hard blocks; users cannot bypass the CSI-2 controller to access the D-PHY directly for non-CSI-2 applications.

The MIPI TX/RX interface supports the MIPI CSI-2 specification v1.3 and the MIPI D-PHY specification v1.1. It has the following features:

- Programmable data lane configuration supporting 1, 2, or 4 lanes
- High-speed mode supports up to 1.5 Gbps data rates per lane
- Operates in continuous and non-continuous clock modes
- 64 bit pixel interface for cameras
- Supports Ultra-Low Power State (ULPS)

Table 42: MIPI Supported Data Types

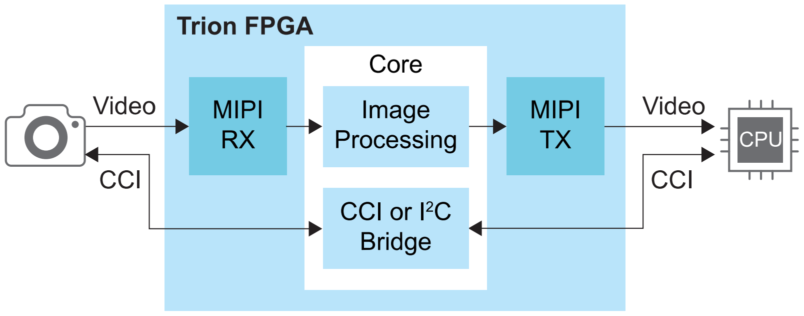
Supported Data Type	Format
RAW	RAW6, RAW7, RAW8, RAW10, RAW12, RAW14
YUV	YUV420 8-bit (legacy), YUV420 8-bit, YUV420 10-bit, YUV420 8-bit (CSPS), YUV420 10-bit (CSPS), YUV422 8-bit, YUV422 10-bit
RGB	RGB444, RGB555, RGB565, RGB666, RGB888

[7] Source: MIPI Alliance https://www.mipi.org/specifications/csi-2

Supported Data Type	Format
User Defined	8 bit format

With more than one MIPI TX and RX blocks, Trion® FPGAs support a variety of video applications.

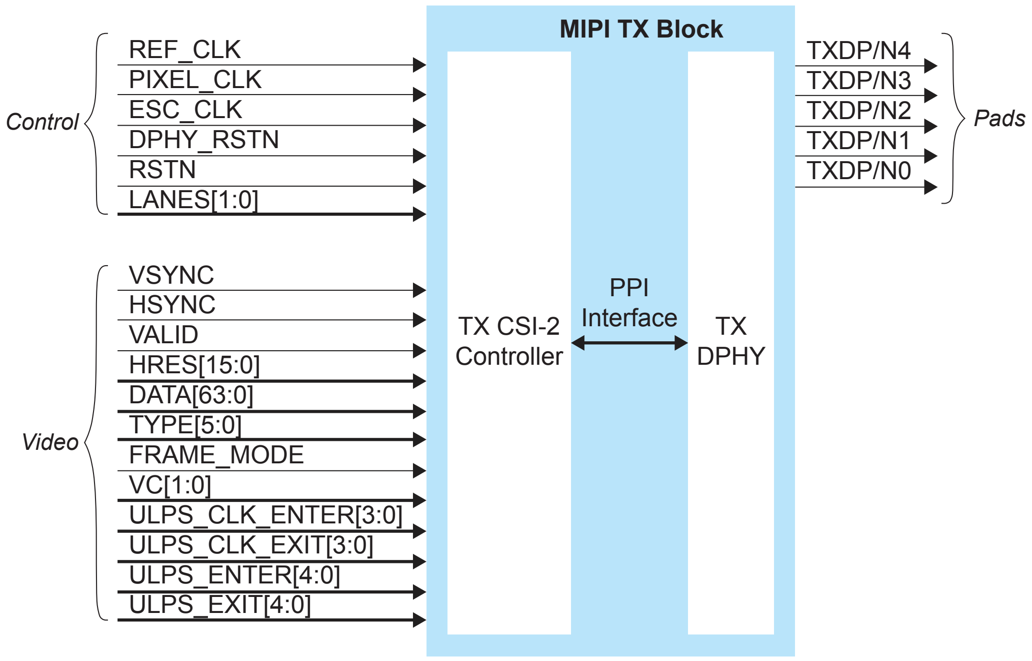
Figure 23: MIPI Example System



MIPI TX

The MIPI TX is a transmitter interface that translates video data from the Trion® core into packetized data sent over the HSSI interface to the board. Five high-speed differential pin pairs (four data, one clock), each of which represent a lane, connect to the board. Control and video signals connect from the MIPI interface to the core.

Figure 24: MIPI TX x4 Block Diagram



The control signals determine the clocking and how many transceiver lanes are used. All control signals are required except the two reset signals. The reset signals are optional, however, you must use both signals or neither.

The MIPI block requires an escape clock (`ESC_CLK`) for use when the MIPI interface is in escape (low-power) mode, which runs between 11 and 20 MHz.

Note: Efinix recommends that you set the escape clock frequency as close to 20 MHz as possible.

The video signals receive the video data from the core. The MIPI interface block encodes is and sends it out through the MIPI D-PHY lanes.

Figure 25: MIPI TX Interface Block Diagram

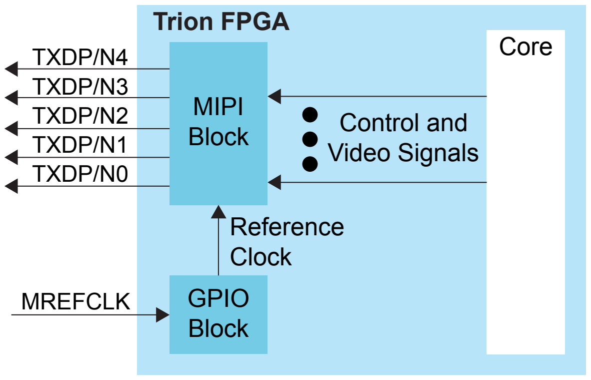


Table 43: MIPI TX Control Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Description
REF_CLK	Input	N/A	Reference clock for the internal MIPI TX PLL used to generate the transmitted data. The FPGA has a dedicated GPIO resource (MREFCLK) that you must configure to provide the reference clock. All of the MIPI TX blocks share this resource. The frequency is set using Interface Designer configuration options.
PIXEL_CLK	Input	N/A	Clock used for transferring data from the core to the MIPI TX block. The frequency is based on the number of lanes and video format. Refer to **Understanding the RX and TX Pixel Clock** on page 68.
ESC_CLK	Input	N/A	Slow clock for escape mode (11 - 20 MHz).
DPHY_RSTN	Input	N/A	(Optional) Reset for the D-PHY logic, active low. Reset with the controller. See **MIPI Reset Timing**.
RSTN	Input	N/A	(Optional) Reset for the CSI-2 controller logic, active low. Typically, you reset the controller with the PHY (see **MIPI Reset Timing**). However, when dynamically changing the horizontal resolution, you only need to trigger RSTN (see **TX Requirements for Dynamically Changing the Horizontal Resolution**).
LANES[1:0]	Input	PIXEL_CLK	Determines the number of lanes enabled. Can only be changed during reset. 00: lane 0 01: lanes 0 and 1 11: all lanes

Table 44: MIPI TX Video Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Description
VSYNC	Input	PIXEL_CLK	Vertical sync.
HSYNC	Input	PIXEL_CLK	Horizontal sync.
VALID	Input	PIXEL_CLK	Valid signal.
HRES[15:0]	Input	PIXEL_CLK	Horizontal resolution. Can only be changed when VSYNC is low, and should be stable for at least one TX pixel clock cycle before VSYNC goes high.
DATA[63:0]	Input	PIXEL_CLK	Video data; the format depends on the data type. New data arrives on every pixel clock.
TYPE[5:0]	Input	PIXEL_CLK	Video data type. Can only be changed when HSYNC is low, and should be stable for at least one TX pixel clock cycle before HSYNC goes high.
FRAME_MODE	Input	PIXEL_CLK	Selects frame format. [8] 0: general frame 1: accurate frame Can only be changed during reset.
VC[1:0]	Input	PIXEL_CLK	Virtual channel (VC). Can only be changed when VSYNC is low, and should be stable at least one TX pixel clock cycle before VSYNC goes high.
ULPS_CLK_ENTER	Input	PIXEL_CLK	Place the clock lane into ULPS mode. Should not be active at the same time as ULPS_CLK_EXIT. Each high pulse should be at least 5 μs.
ULPS_CLK_EXIT	Input	PIXEL_CLK	Remove clock lane from ULPS mode. Should not be active at the same time as ULPS_CLK_ENTER. Each high pulse should be at least 5 μs.
ULPS_ENTER[3:0]	Input	PIXEL_CLK	Place the data lane into ULPS mode. Should not be active at the same time as ULPS_EXIT[3:0]. Each high pulse should be at least 5 μs.
ULPS_EXIT[3:0]	Input	PIXEL_CLK	Remove the data lane from ULPS mode. Should not be active at the same time as ULPS_ENTER[3:0]. Each high pulse should be at least 5 μs.

Table 45: MIPI TX Pads

Pad	Direction	Description
TXDP[4:0]	Output	MIPI transceiver P pads.
TXDN[4:0]	Output	MIPI transceiver N pads.

[8] Refer to the MIPI Camera Serial Interface 2 (MIPI CSI-2) for more information about frame formats.

Table 46: MIPI TX Settings in Efinity® Interface Designer

Tab	Parameter	Choices	Notes
Base	PHY Frequency (MHz)	80.00 - 1500.00	Choose one of the possible PHY frequency values.
	Frequency (reference clock)	6, 12, 19.2, 25, 26, 27, 38.4, or 52 MHz	Reference clock frequency.
	Enable Continuous PHY Clocking	On or Off	Turns continuous clock mode on or off.
Control	Escape Clock Pin Name	User defined	
	Invert Escape Clock	On or Off	
	Pixel Clock Pin Name	User defined	
	Invert Pixel Clock	On or Off	
Lane Mapping	TXD0, TXD1, TXD2, TXD3, TXD4	clk, data0, data1, data2, or data3	Map the physical lane to a clock or data lane.
	Clock Timer		
Timing	$T_{CLK-POST}$ $T_{CLK-TRAIL}$ $T_{CLK-PREPARE}$ $T_{CLK-ZERO}$	Varies depending on the PHY frequency	Changes the MIPI transmitter timing parameters per the DPHY specification. Refer to D-PHY Timing Parameters on page 67.
	Escape Clock Frequency (MHz)	User defined	Specify a number between 11 and 20 MHz.
	$T_{CLK-PRE}$	Varies depending on the escape clock frequency	Changes the MIPI transmitter timing parameters per the DPHY specification. Refer to D-PHY Timing Parameters on page 67.
	Data Timer		
	$T_{HS-PREPARE}$ $T_{HS-ZERO}$ $T_{HS-PTRAIL}$	Varies depending on the PHY frequency	Changes the MIPI transmitter timing parameters per the DPHY specification. Refer to D-PHY Timing Parameters on page 67.

MIPI TX Video Data TYPE[5:0] Settings

The video data type can only be changed when HSYNC is low.

Table 47: MIPI TX TYPE[5:0]

TYPE[5:0]	Data Type	Pixel Data Bits per Pixel Clock	Pixels per Clock	Bits per Pixel	Maximum Data Pixels per Line
0x20	RGB444	48	4	12	2,880
0x21	RGB555	60	4	15	2,880
0x22	RGB565	64	4	16	2,880
0x23	RGB666	54	3	18	2,556
0x24	RGB888	48	2	24	1,920
0x28	RAW6	60	10	6	7,680
0x29	RAW7	56	8	7	6,576
0x2A	RAW8	64	8	8	5,760
0x2B	RAW10	60	6	10	4,608

TYPE[5:0]	Data Type	Pixel Data Bits per Pixel Clock	Pixels per Clock	Bits per Pixel	Maximum Data Pixels per Line
0x2C	RAW12	60	5	12	3,840
0x2D	RAW14	56	4	14	3,288
0x18	YUV420 8 bit	Odd line: 64 Even line: 64	Odd line: 8 Even line: 4	Odd line: 8 Even line: 8, 24	2,880
0x19	YUV420 10 bit	Odd line: 60 Even line: 40	Odd line: 6 Even line: 2	Odd line: 10 Even line: 10, 30	2,304
0x1A	Legacy YUV420 8 bit	48	4	8, 16	3,840
0x1C	YUV420 8 bit (CSPS)	Odd line: 64 Even line: 64	Odd line: 8 Even line: 4	Odd line: 8 Even line: 8, 24	2,880
0x1D	YUV420 10 bit (CSPS)	Odd line: 60 Even line: 40	Odd line: 6 Even line: 2	Odd line: 10 Even line: 10, 30	2,304
0x1E	YUV422 8 bit	64	4	8, 24	2,880
0x1F	YUV422 10 bit	40	2	10, 30	2,304
0x30 - 37	User defined 8 bit	64	8	8	5,760

MIPI TX Video Data DATA[63:0] Formats

The format depends on the data type. New data arrives on every pixel clock.

Table 48: RAW6 (10 Pixels per Clock)

63 60	59 54	53 48	47 42	41 36	35 30	29 24	23 18	17 12	11 6	5 0
0	Pixel 10	Pixel 9	Pixel 8	Pixel 6	Pixel 6	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 49: RAW7 (8 Pixels per Clock)

63 56	55 49	48 42	41 35	34 28	27 21	20 14	13 7	6 0
0	Pixel 8	Pixel 7	Pixel 6	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 50: RAW8 and User Defined (8 Pixels per Clock)

63 54	53 48	47 40	39 32	31 24	23 16	15 8	7 0
Pixel 8	Pixel 7	Pixel 6	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 51: RAW10 (6 Pixels per Clock)

63 60	59 50	49 40	39 30	29 20	19 10	9 0
0	Pixel 6	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 52: RAW12 (5 Pixels per Clock)

63 60	59 48	47 36	35 24	23 12	11 0
0	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 53: RAW14 (4 Pixels per Clock)

63 56	55 42	41 28	27 14	13 0
0	Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 54: RGB444 (4 Pixels per Clock)

63 48	47 36	35 24	23 12	11 0
0	Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 55: RGB555 (4 Pixels per Clock)

63 60	59 45	44 30	29 15	14 0
0	Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 56: RGB565 (4 Pixels per Clock)

63 48	47 32	31 16	15 0
Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 57: RGB666 (3 Pixels per Clock)

63 54	53 36	35 18	17 0
0	Pixel 3	Pixel 2	Pixel 1

Table 58: RGB888 (2 Pixels per Clock)

63	4847	2423 0
0	Pixel 2	Pixel 1

Table 59: YUV420 8 bit Odd Line (8 Pixels per Clock), Even Line (4 Pixels per Clock)

63	5655	4847	4039	3231	2423	1615	8 7 0
Odd Lines							
Pixel 8 Y8	Pixel 7 Y7	Pixel 6 Y6	Pixel 5 Y5	Pixel 4 Y4	Pixel 3 Y3	Pixel 2 Y2	Pixel 1 Y1
Even Lines							
Pixel 4	Pixel 3			Pixel 2	Pixel 1		
Y4	V3	Y3	U3	Y2	V1	Y1	U1

Table 60: Legacy YUV420 8 bit (4 Pixels per Clock)

63	4847	4039	3231	2423	1615	8 7 0
0	Pixel 4	Pixel 3		Pixel 2	Pixel 1	
Odd Lines	Y4	Y3	U3	Y2	Y1	U1
Even Lines	Y4	Y3	V3	Y2	Y1	V1

Table 61: YUV420 10 bit Odd Line (6 Pixels per Clock) Even Line (2 Pixels per Clock)

63	6059	5049	4039	3029	2019	10 9 0
Odd Lines						
0	Pixel 6 Y6	Pixel 5 Y5	Pixel 4 Y4	Pixel 3 Y3	Pixel 2 Y2	Pixel 1 Y1
Even Lines						
0			Pixel 2	Pixel 1		
			Y2	V1	Y1	U1

Table 62: YUV422 8 bit (4 Pixels per Clock)

63	5655	4847	4039	3231	2423	1615	8 7 0
Pixel 4	Pixel 3			Pixel 2	Pixel 1		
Y4	V3	Y3	U3	Y2	V1	Y1	U1

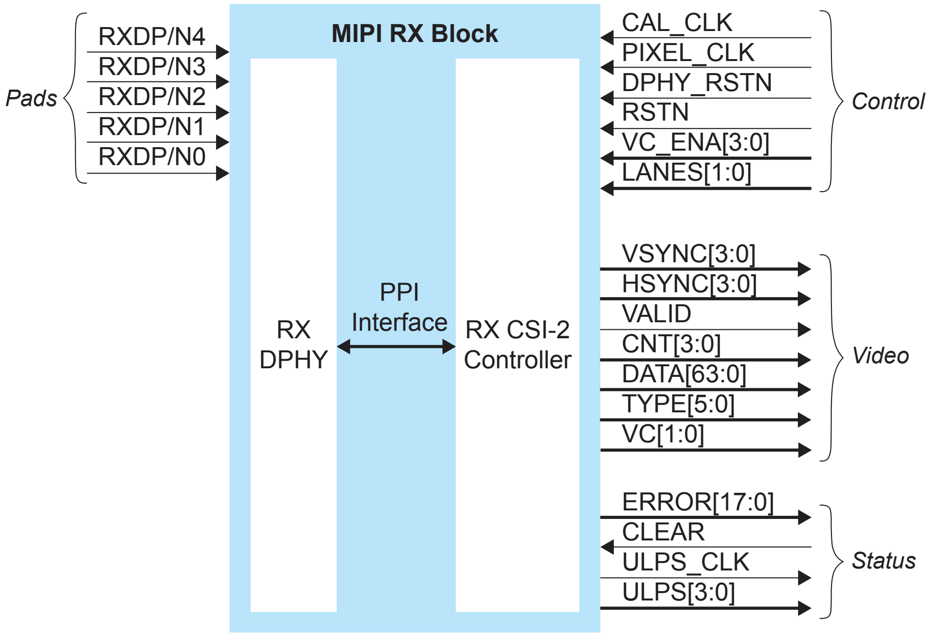
Table 63: YUV422 10 bit (2 Pixels per Clock)

63	4039	3029	2019	10 9 0
0	Pixel 2	Pixel 1		
	Y2	V1	Y1	U1

MIPI RX

The MIPI RX is a receiver interface that translates HSSI signals from the board to video data in the Trion® core. Five high-speed differential pin pairs (one clock, four data), each of which represent a lane, connect to the board. Control, video, and status signals connect from the MIPI interface to the core.

Figure 26: MIPI RX x4 Block Diagram



The control signals determine the clocking, how many transceiver lanes are used, and how many virtual channels are enabled. All control signals are required except the two reset signals. The reset signals are optional, however, you must use both signals or neither.

The video signals send the decoded video data to the core. All video signals must fully support the MIPI standard.

The status signals provide optional status and error information about the MIPI RX interface operation.

Figure 27: MIPI RX Interface Block Diagram

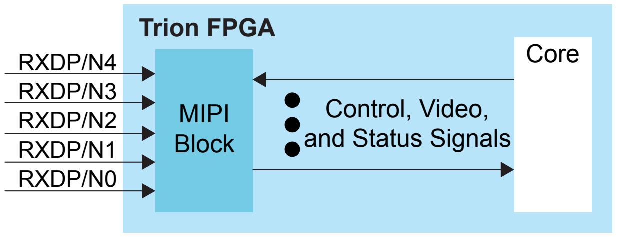


Table 64: MIPI RX Control Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Notes
CAL_CLK	Input	N/A	Used for D-PHY calibration; must be between 80 and 120 MHz.
PIXEL_CLK	Input	N/A	Clock used for transferring data to the core from the MIPI RX block. The frequency based on the number of lanes and video format. Refer to **Understanding the RX and TX Pixel Clock** on page 68.
DPHY_RSTN	Input	N/A	(Optional) Reset for the D-PHY logic, active low. Must be used if RSTN is used. See **MIPI Reset Timing**.
RSTN	Input	N/A	(Optional) Reset for the CSI-2 controller logic, active low. Must be used if DPHY_RSTN is used. See **MIPI Reset Timing**.
VC_ENA[3:0]	Input	PIXEL_CLK	Enables different VC channels by setting their index high.
LANES[1:0]	Input	PIXEL_CLK	Determines the number of lanes enabled: 00: lane 0 01: lanes 0 and 1 11: all lanes Can only be set during reset.

Table 65: MIPI RX Video Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Notes
VSYNC[3:0]	Output	PIXEL_CLK	Vsync bus. High if vsync is active for this VC.
HSYNC[3:0]	Output	PIXEL_CLK	Hsync bus. High if hsync is active for this VC
VALID	Output	PIXEL_CLK	Valid signal.
CNT[3:0]	Output	PIXEL_CLK	Number of valid pixels contained in the pixel data.
DATA[63:0]	Output	PIXEL_CLK	Video data, format depends on data type. New data every pixel clock.
TYPE[5:0]	Output	PIXEL_CLK	Video data type.
VC[1:0]	Output	PIXEL_CLK	Virtual channel (VC).

Table 66: MIPI RX Status Signals (Interface to FPGA Fabric)

Signal	Direction	Signal Interface	Clock Domain	Notes
ERROR[17:0]	Output	IN	PIXEL_CLK	Error bus register. Refer to **Table 67: MIPI RX Error Signals (ERROR[17:0])** on page 62 for details.
CLEAR	Input	OUT	PIXEL_CLK	Reset the error registers.
ULPS_CLK	Output	IN	PIXEL_CLK	High when the clock lane is in the Ultra-Low-Power State (ULPS).
ULPS[3:0]	Output	IN	PIXEL_CLK	High when the lane is in the ULPS mode.

Table 67: MIPI RX Error Signals (ERROR[17:0])

Bit	Name	Description
0	ERR_ESC	Escape Entry Error. Asserted when an unrecognized escape entry command is received.
1	CRC_ERROR_VC0	CRC Error VC0. Set to 1 when a checksum error occurs.
2	CRC_ERROR_VC1	CRC Error VC1. Set to 1 when a checksum error occurs.
3	CRC_ERROR_VC2	CRC Error VC2. Set to 1 when a checksum error occurs.
4	CRC_ERROR_VC3	CRC Error VC3. Set to 1 when a checksum error occurs.
5	HS_RX_TIMEOUT_ERR	HS RX Timeout Error. The protocol should time out when no EoT is received within a certain period in HS RX mode.
6	ECC_1BIT_ERROR	ECC Single Bit Error. Set to 1 when there is a single bit error.
7	ECC_2BIT_ERROR	ECC 2 Bit Error. Set to 1 if there is a 2 bit error in the packet.
8	ECCBIT_ERROR	ECC Error. Asserted when an error exists in the ECC.
9	ECC_NO_ERROR	ECC No Error. Asserted when an ECC is computed with a result zero. This bit is high when the receiver is receiving data correctly.
10	FRAME_SYNC_ERROR	Frame Sync Error. Asserted when a frame end is not paired with a frame start on the same virtual channel.
11	INVLD_PKT_LEN	Invalid Packet Length. Set to 1 if there is an invalid packet length.
12	INVLD_VC	Invalid VC ID. Set to 1 if there is an invalid CSI VC ID.
13	INVALID_DATA_TYPE	Invalid Data Type. Set to 1 if the received data is invalid.
14	ERR_FRAME	Error In Frame. Asserted when VSYNC END received when CRC error is present in the data packet.
15	CONTROL_ERR	Control Error. Asserted when an incorrect line state sequence is detected.
16	SOT_ERR	Start-of-Transmission (SoT) Error. Corrupted high-speed SoT leader sequence while proper synchronization can still be achieved.
17	SOT_SYNC_ERR	SoT Synchronization Error. Corrupted high-speed SoT leader sequence while proper synchronization cannot be expected.

Table 68: MIPI RX Pads

Pad	Direction	Description
RXDP[4:0]	Input	MIPI transceiver P pads.
RXDN[4:0]	Input	MIPI transceiver N pads.

Table 69: MIPI RX Settings in Efinity® Interface Designer

Tab	Parameter	Choices	Notes
Control	DPHY Calibration Clock Pin Name	User defined	
	Invert DPHY Calibration Clock	On or Off	
	Pixel Clock Pin Name	User defined	
	Invert Pixel Clock	On or Off	
Status	Enable Status	On or Off	Indicate whether you want to use the status pins.
Lane Mapping	RXD0, RXD1, RXD2, RXD3, RXD4	clk, data0, data1, data2, or data3	Map the physical lane to a clock or data lane.
	Swap P&N Pin	On or Off	Reverse the P and N pins for the physical lane.
Timing	Calibration Clock Freq (MHz)	User defined	Specify a number between 80 and 120 MHz.
	Clock Timer ($T_{CLK\text{-}SETTLE}$)	40 - 2,590 ns	Changes the MIPI receiver timing parameters per the DPHY specification. Refer to D-PHY Timing Parameters on page 67.
	Data Timer ($T_{HS\text{-}SETTLE}$)	40 - 2,590 ns	Changes the MIPI receiver timing parameters per the DPHY specification. Refer to D-PHY Timing Parameters on page 67.

MIPI RX Video Data TYPE[5:0] Settings

The video data type can only be changed when HSYNC is low.

Table 70: MIPI RX TYPE[5:0]

TYPE[5:0]	Data Type	Pixel Data Bits per Pixel Clock	Pixels per Clock	Bits per Pixel	Maximum Data Pixels per Line
0x20	RGB444	48	4	12	2,880
0x21	RGB555	60	4	15	2,880
0x22	RGB565	64	4	16	2,880
0x23	RGB666	54	3	18	2,556
0x24	RGB888	48	2	24	1,920
0x28	RAW6	48	8	6	7,680
0x29	RAW7	56	8	7	6,576
0x2A	RAW8	64	8	8	5,760
0x2B	RAW10	40	4	10	4,608
0x2C	RAW12	48	4	12	3,840
0x2D	RAW14	56	4	14	3,288

TYPE[5:0]	Data Type	Pixel Data Bits per Pixel Clock	Pixels per Clock	Bits per Pixel	Maximum Data Pixels per Line
0x18	YUV420 8 bit	Odd line: 64 Even line: 64	Odd line: 8 Even line: 4	Odd line: 8 Even line: 8, 24	2,880
0x19	YUV420 10 bit	Odd line: 40 Even line: 40	Odd line: 4 Even line: 2	Odd line: 10 Even line: 10, 30	2,304
0x1A	Legacy YUV420 8 bit	48	4	8, 16	3,840
0x1C	YUV420 8 bit (CSPS)	Odd line: 64 Even line: 64	Odd line: 8 Even line: 4	Odd line: 8 Even line: 8, 24	2,880
0x1D	YUV420 10 bit (CSPS)	Odd line: 40 Even line: 40	Odd line: 4 Even line: 2	Odd line: 10 Even line: 10, 30	2,304
0x1E	YUV422 8 bit	64	4	8, 24	2,880
0x1F	YUV422 10 bit	40	2	10, 30	2,304
0x30 - 37	User defined 8 bit	64	8	8	5,760

MIPI RX Video Data DATA[63:0] Formats

The format depends on the data type. New data arrives on every pixel clock.

Table 71: RAW6 (8 Pixels per Clock)

63 48	47 42	41 36	35 30	29 24	23 18	17 12	11 6	5 0
0	Pixel 8	Pixel 7	Pixel 6	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 72: RAW7 (8 Pixels per Clock)

63 56	55 49	48 42	41 35	34 28	27 21	20 14	13 7	6 0
0	Pixel 8	Pixel 7	Pixel 6	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 73: RAW8 (8 Pixels per Clock)

63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
Pixel 8	Pixel 7	Pixel 6	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 74: RAW10 (4 Pixels per Clock)

63 40	39 30	29 20	19 10	9 0
0	Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 75: RAW12 (4 Pixels per Clock)

63 48	47 36	35 24	23 12	11 0
0	Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 76: RAW14 (4 Pixels per Clock)

63 56	55 42	41 28	27 14	13 0
0	Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 77: RGB444 (4 Pixels per Clock)

63 48	47 36	35 24	23 12	11 0
0	Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 78: RGB555 (4 Pixels per Clock)

63 60	59 45	44 30	29 15	14 0
	Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 79: RGB565 (4 Pixels per Clock)

63 48	47 32	31 16	15 0
Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 80: RGB666 (3 Pixels per Clock)

63 54	53 36	35 18	17 0
0	Pixel 3	Pixel 2	Pixel 1

Table 81: RGB888 (2 Pixels per Clock)

63 48	47 24	23 0
0	Pixel 2	Pixel 1

Table 82: YUV420 8 bit Odd Line (8 Pixels per Clock), Even Line (4 Pixels per Clock)

63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
Odd Lines							
Pixel 8 Y8	Pixel 7 Y7	Pixel 6 Y6	Pixel 5 Y5	Pixel 4 Y4	Pixel 3 Y3	Pixel 2 Y2	Pixel 1 Y1
Even Lines							
Pixel 4	Pixel 3			Pixel 2	Pixel 1		
Y4	U3	Y3	V3	Y2	U1	Y1	V1

Table 83: Legacy YUV420 8 bit (4 Pixels per Clock)

63 48	47 40	39 32	31 24	23 16	15 8	7 0
0	Pixel 4	Pixel 3		Pixel 2	Pixel 1	
Odd Lines	Y4	U3	Y3	Y2	U1	Y1
Even Lines	Y4	V3	Y3	Y2	V1	Y1

Table 84: YUV420 10 bit Odd Line (4 Pixels per Clock), Even Line (2 Pixels per Clock)

63 40	39 30	29 20	19 10	9 0
Odd Lines				
0	Pixel 4 Y4	Pixel 3 Y3	Pixel 2 Y2	Pixel 1 Y1
Even Lines				
0	Pixel 1	Pixel 2	Pixel 1	
	V1	Y2	U1	Y1

Table 85: YUV422 8 bit (4 Pixels per Clock)

63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
Pixel 4	Pixel 3			Pixel 2	Pixel 1		
Y4	V3	Y3	U3	Y2	V1	Y1	U1

Table 86: YUV422 10 bit (2 Pixels per Clock)

63 40	39 30	29 20	19 10	9 0
0	Pixel 1	Pixel 2	Pixel 1	
	V1	Y2	U1	Y1

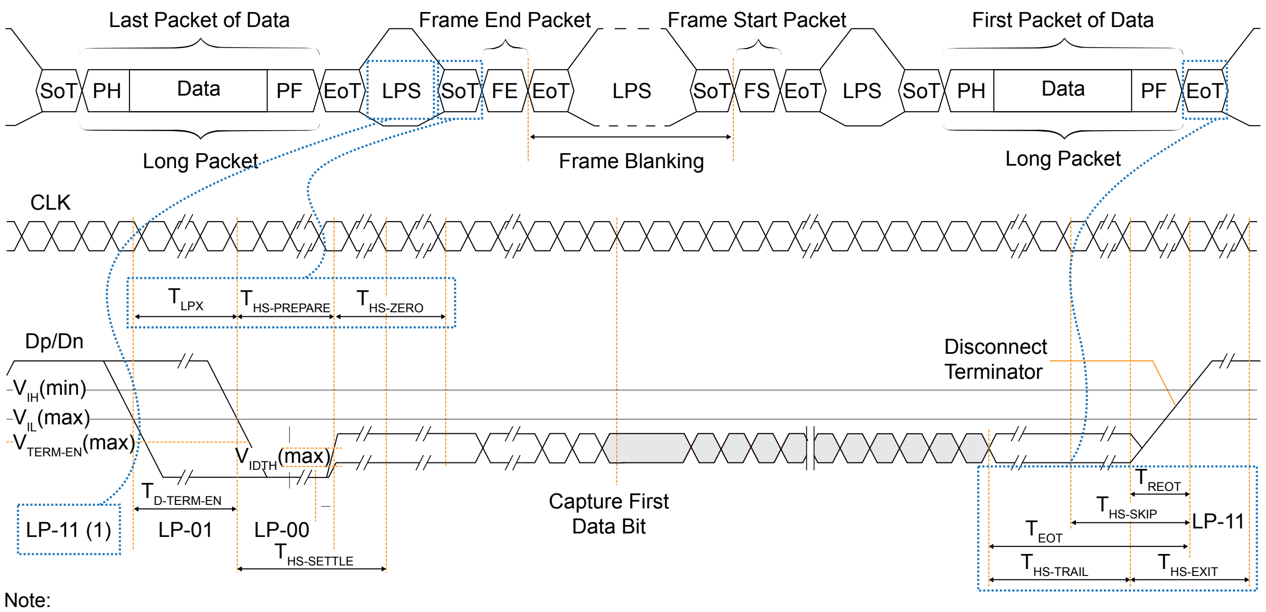
D-PHY Timing Parameters

During CSI-2 data transmission, the MIPI D-PHY alternates between low power mode and high-speed mode. The D-PHY specification defines timing parameters to facilitate the correct hand-shaking between the MIPI TX and MIPI RX during mode transitions.

You set the timing parameters to correspond to the specifications of your hardware in the Efinity® Interface Designer.

- *RX parameters*—$T_{CLK-SETTLE}$, $T_{HS-SETTLE}$ (see Table 64: MIPI RX Control Signals (Interface to FPGA Fabric) on page 61)
- *TX parameters*—$T_{CLK-POST}$, $T_{CLK-TRAIL}$, $T_{CLK-PREPARE}$, $T_{CLK-ZERO}$, $T_{CLK-PRE}$, $T_{HS-PREPARE}$, $T_{HS-ZERO}$, $T_{HS-TRAIL}$ (see Table 46: MIPI TX Settings in Efinity Interface Designer on page 56)

Figure 28: High-Speed Data Transmission in Bursts Waveform



Note:
1. To enter high-speed mode, the D-PHY goes through states LP-11, LP-01, and LP-00. The D-PHY generates LP-11 to exit high-speed mode.

Figure 29: Switching the Clock Lane between Clock Transmission and Low Power Mode Waveform

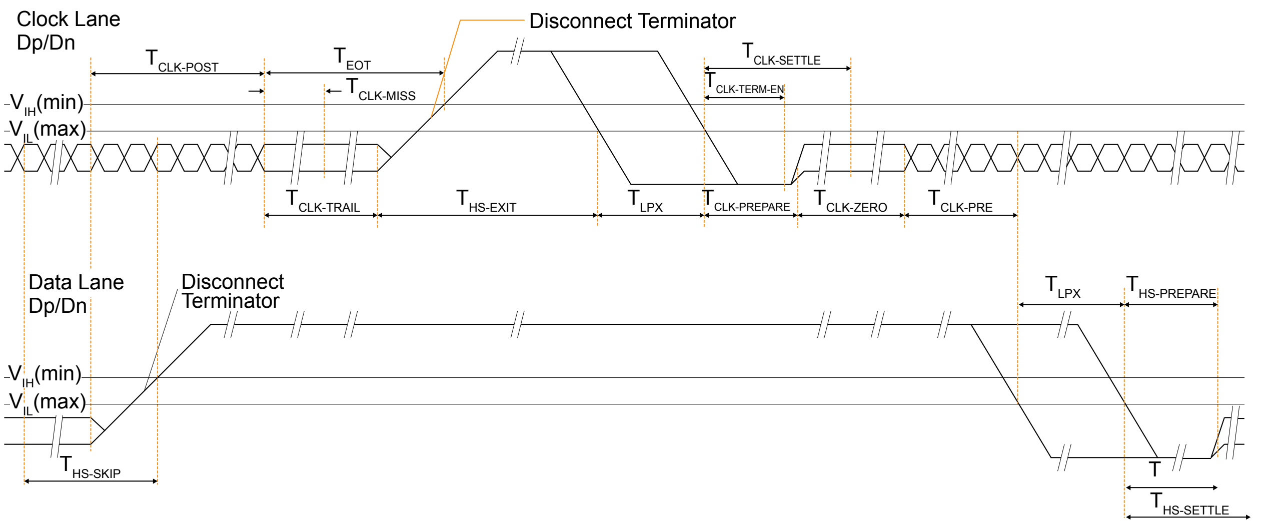


Table 87: D-PHY Timing Specifications

Parameter	Description	Min	Typ	Max	Unit
$T_{CLK-POST}$	Time that the transmitter continues to send HS clock after the last associated Data Lane has transitioned to LP Mode. Interval is defined as the period from the end of $T_{HS-TRAIL}$ to the beginning of $T_{CLK-TRAIL}$.	60 ns + 52*UI	–	–	ns
$T_{CLK-PRE}$	Time that the HS clock shall be driven by the transmitter prior to any associated Data Lane beginning the transition from LP to HS mode.	8	–	–	UI
$T_{CLK-PREPARE}$	Time that the transmitter drives the Clock Lane LP-00 Line state immediately before the HS-0 Line state starting the HS transmission.	38	–	95	ns
$T_{CLK-SETTLE}$	Time interval during which the HS receiver should ignore any Clock Lane HS transitions, starting from the beginning of $T_{CLK-PREPARE}$.	95	–	300	ns
$T_{CLK-TRAIL}$	Time that the transmitter drives the HS-0 state after the last payload clock bit of a HS transmission burst.	60	–	–	ns
$T_{CLK-PREPARE}$ + $T_{CLK-ZERO}$	$T_{CLK-PREPARE}$ + time that the transmitter drives the HS-0 state prior to starting the Clock.	300	–	–	ns
$T_{HS-PREPARE}$	Time that the transmitter drives the Data Lane LP-00 Line state immediately before the HS-0 Line state starting the HS transmission	40 ns + 4*UI	–	85 ns + 6*UI	ns
$T_{HS-SETTLE}$	Time interval during which the HS receiver shall ignore any Data Lane HS transitions, starting from the beginning of $T_{HS-PREPARE}$. The HS receiver shall ignore any Data Lane transitions before the minimum value, and the HS receiver shall respond to any Data Lane transitions after the maximum value.	85 ns + 6*UI	–	145 ns + 10*UI	ns
$T_{HS-TRAIL}$	Time that the transmitter drives the flipped differential state after last payload data bit of a HS transmission burst	max(n*8*UI, 60 ns + n*4*UI)	–	–	ns
T_{LPX}	Transmitted length of any Low-Power state period	50	–	–	ns
$T_{HS-PREPARE}$ + $T_{HS-ZERO}$	$T_{HS-PREPARE}$ + time that the transmitter drives the HS-0 state prior to transmitting the Sync sequence.	145 ns + 10*UI	–	–	ns

Understanding the RX and TX Pixel Clock

In a MIPI system, the pixel clock is the clock used to transfer the video data to or from the MIPI controller. This document calls it the *system pixel clock*. The system pixel clock is related to the video resolution. If you are using a standard monitor, you can simply look up the clock specification in the SMPTE/CEA or VESA standards. Alternatively, you can calculate the clock you need.

Generally speaking, you calculate the system pixel clock frequency using the following equation:

Pixel Clock Frequency = *Total Horizontal Samples* x *Total Vertical Lines* x *Refresh Rate*[9]

where the *Total Horizontal Samples* and *Total Vertical Lines* include the blanking period.

In the Interface Designer, the MIPI TX and RX interfaces also have RX and TX pixel clocks. These clocks are not the same as the system pixel clock, however, they are related. These pixel clocks take into account both the system pixel clock and the video data type.

The RX and TX pixel clocks must be equal to or faster than the system pixel clock divided by the number of pixels processed by the MIPI interface each clock cycle. The number of pixels processed per clock depends on the video data type.

For example, if the system pixel clock is running at 150 MHz and using the RGB444 RX data type, which processes 4 pixels per clock, the RX pixel clock must be at least 37.5 MHz.

Efinix provides a MIPI utility that you can use to calculate the TX and RX pixel clock frequencies that work with the video data type. Refer to Using the MIPI Utility.

Video Data Type

The video data type includes the color mode (RAW, RGB, and YUV) and the data format, which together determine the amount of video transmitted every pixel clock (that is, the bandwidth). The overal system bandwidth is simply the system pixel clock times the number of bits of video data transferred each clock cycle.

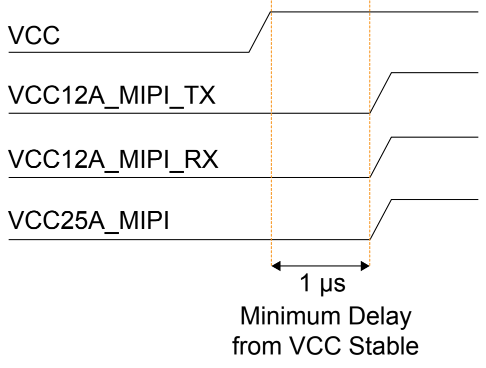
Power Up Sequence

The MIPI block has four power supplies:

- **VCC**—Digital supply voltage
- **VCC25A_MIPI**—2.5 V analog supply voltage
- **VCC12A_MIPI_TX**—1.2 V analog voltage supply to the MIPI TX
- **VCC12A_MIPI_RX**—1.2 V analog voltage supplies to the MIPI RX

When powering up the FPGA, VCC should power up and stabilize before the MIPI analog supplies power up.

Figure 30: MIPI Power Up Sequence



[9] The *Refresh Rate* is also called the frame rate or vertical frequency.

Using the MIPI Block

You use the MIPI TX and RX blocks to configure the hard MIPI interface. You can add up to two MIPI TX and up to two MIPI RX blocks.

Note: The Efinity® software v2019.1 and later supports the MIPI interface block.

MIPI TX

The MIPI TX Block Editor organizes the settings into tabs. Most settings are simply naming the MIPI signals for your design and specifying timing parameters.

Table 88: Base Tab Settings

The Base tab is where you set the instance name, choose a resource, and make clock settings.

Setting	Choices	Notes
Instance Name	User defined	Specify an instance name.
MIPI TX Resource	MIPI_TX0 or MIPI_TX1	Choose which resource to instantiate.
PHY Frequency (MHz)	80 to 1500 MHz	Choose the speed at which to run the D-PHY.
Reference Clock Frequency (MHz)	6.00, 12.00. 19.00, 25.00, 26.00, 27.00, 38.00, 52.00	The software automatically assigns a GPIO resource for the reference clock. You must add a GPIO block in clock output mode and assign it this resource. Both MIPI resources share the same reference clock. Therefore, you must use the same frequency setting for both instances.
Enable Continuous PHY Clocking	On or Off	

- **Control Tab**—Specify names for control signals. The DPHY and CSI-2 resets are optional. Leave **DPHY Reset Pin Name** and **CSI-2 Reset Pin Name** blank if you do not want to use the resets. You must use both resets or neither.
- **Video Tab**—Specify names for video signals.
- **Video - ULPS Mode Tab**—The MIPI TX block supports the Ultra-Low Power State (ULPS) for the data and clock lanes. In this mode, the lane goes to sleep and does not transmit data. The MIPI block consumes almost no power in ULPS mode. If you want to use ULPS mode, specify the names of the ULPS enter and exit signals for the clock and data lanes.

  Note: The MIPI CSI-2 controller automatically uses escape mode and low-power data transmission for low-power operation. You do not need to enable these modes.

- **Lane Mapping Tab**—The MIPI TX block supports 4 data lanes and 1 clock lane. In this tab you choose which lane to associate with the MIPI pad. Select a name from the drop-down list. The lane mapping must be unique, which the software enforces.
- **Timing Tab**—In this tab you specify the timing parameters for the clock and data timers.

MIPI RX

The MIPI TX Block Editor organizes the settings into tabs. Most settings are simply naming the MIPI signals for your design and specifying timing parameters.

- **Base Tab**—Specify an instance name and choose a MIPI RX resource. Choose **MIPI_RX0** or **MIPI_RX1**.

- **Control Tab**—Specify names for control signals.
- **Video Tab**—Specify names for video signals.
- **Status Tab**—You can choose to enable status signals. Turn on **Enable Status** and specify the signal names.
- **Lane Mapping Tab**—The MIPI TR block supports 4 data lanes and 1 clock lane. In this tab you choose which lane to associate with the MIPI pad. Select a name from the drop-down list. The lane mapping must be unique, which the software enforces.
- **Timing Tab**—In this tab you specify the timing parameters for the clock and data timers, as well as the calibration clock frequency.

Chapter 8

PLL Interface

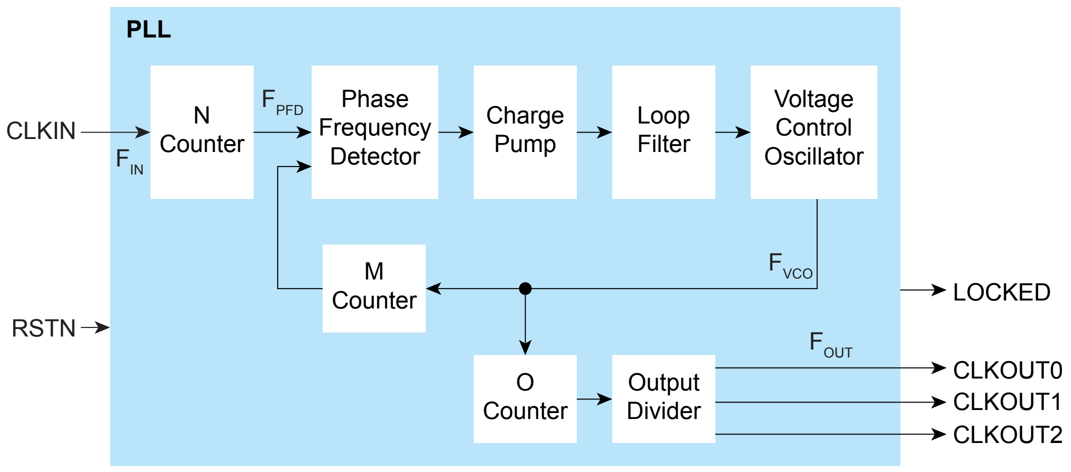
Contents:

The following sections describe the simple PLL and how to use it. Refer to the Package/Interface Support Matrix on page 8 to find out if your FPGA supports the simple PLL.

About the Simple PLL Interface

The Trion has 1 PLL to synthesize clock frequencies. The PLL's reference clock input comes from a dedicated GPIO's alternate input pin. The PLL consists of a pre-divider counter (N counter), a feedback multiplier counter (M counter), post-divider counter (O counter), and an output divider per clock output.

Figure 31: Trion PLL Block Diagram



The counter settings define the PLL output frequency:	where:
$F_{PFD} = F_{IN} / N$ $F_{VCO} = F_{PFD} \times M$ $F_{OUT} = F_{VCO} / (O \times \text{Output divider})$	F_{VCO} is the voltage control oscillator frequency F_{OUT} is the output clock frequency F_{IN} is the reference clock frequency F_{PFD} is the phase frequency detector input frequency

Note: The reference clock must be between 10 and 50 MHz.
The PFD input must be between 10 and 50 MHz.
The VCO frequency must be between 500 and 1,500 MHz.

Unlike other Trion® FPGAs, the Trion PLL output locks on the *negative* clock edge (not the positive edge). When you are using two or more clock outputs, they are aligned on the falling edge. If the core register receiving the clock is positive edge triggered, Efinix recommends inverting the clock outputs so they are correctly edge aligned.

Figure 32: PLL Output Aligned with Negative Edge

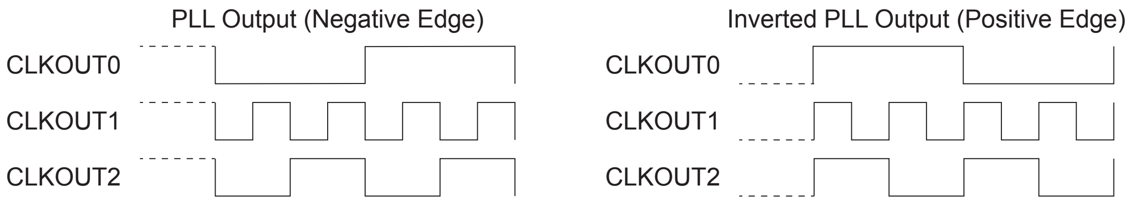

PLL Output (Negative Edge)
Inverted PLL Output (Positive Edge)
CLKOUT0
CLKOUT1
CLKOUT2
CLKOUT0
CLKOUT1
CLKOUT2

Table 89: PLL Pins

Port	Direction	Description
CLKIN	Input	Reference clock. This port is also a GPIO pin; the GPIO pins' alternate function is configured as a reference clock.
RSTN	Input	Active-low PLL reset signal. When asserted, this signal resets the PLL; when de-asserted, it enables the PLL. Connect this signal in your design to power up or reset the PLL. Assert the RSTN pin for a minimum pulse of 10 ns to reset the PLL.
CLKOUT0 CLKOUT1 CLKOUT2	Output	PLL output. The designer can route these signals as input clocks to the core's GCLK network.
LOCKED	Output	Goes high when PLL achieves lock; goes low when a loss of lock is detected. Connect this signal in your design to monitor the lock status. This signal is analog asynchronous.

Table 90: PLL Settings

Configure these settings in the Efinity® Interface Designer.

Setting	Allowed Values	Notes
N counter	1 - 15 (integer)	Pre-divider
M counter	1 - 255 (integer)	Multiplier
O counter	1, 2, 4, 8	Post-divider
Output divider	2, 4, 8, 16, 32, 64, 128, 256	Output divider per output

Using the PLL V1 Block

The Trion T4 and T8 FPGAs have a simple PLL. This PLL is referenced as V1.

Note: In this mode, a specific GPIO block with an alternate connection type must generate the reference clock(s). The PLL Block Summary shows the resource name to use.

1. Add a GPIO block.
2. Enter the instance name.
3. Choose **input** as the mode.
4. Choose **pll_clkin** as the connection type.
5. In the Resource Assigner, assign it to the resource shown in the summary.

You can set up the PLL using the PLL Clock Calculator or manually using the Block Editor.

- In the PLL's **Properties** tab, you specify general settings such as the instance name, PLL resource, clock source, and external clock.
- Click the **Automated Clock Calculation** button to open the PLL Clock Calculator.
- Click the **Manual Configuration** tab to configure the PLL manually.

Using the PLL Clock Calculator

The PLL Clock Calculator provides a graphical way for you to set up the simple PLL block. When you open the calculator, the GUI appears in automatic mode, which provides basic settings. You can:

- Turn signals on or off by clicking the icons (gray x or green arrow) next to the signal.
- Specify the signal names.
- Choose the clock phase.

As you make selections, the calculator determines the values for the pre-divider, multiplier, post divider, and clock dividers that meet your settings. The GUI prompts you if you make selections that are impossible to solve.

In manual mode, the interface displays the PLL's internal block diagram, and provides boxes for you to set the values for the pre-divider, multiplier, post divider, and clock dividers. As you adjust the values, the calculator prompts you if you make settings that result in F_{VCO} values that are out of range or are impossible to solve. When you turn manual mode off, the calculator adjusts the output clock frequencies to match the manual settings. If you have incorrect settings for the pre-divider, multiplier, post divider, and clock dividers, when you turn manual mode off, the calculator adjusts the values to ones that allow a valid solution.

When you are finished using the calculator, click **Finish** to save your settings and close the GUI.

Set up the PLL Manually

Make these settings in the **Manual Configuration** tab.

- Specify general settings such as the PLL resource, reset pin name, and locked pin name.
- Under VCO Frequency, specify the reference clock frequency, multiplier, and pre-divider. The software calculates and displays the resulting VCO frequency. If the VCO is outside of the allowed range, it displays in red.
- Under PLL Frequency, choose the post divider. The software calculates and displays the PLL frequency.
- The simple PLL has three output clocks. Enable the output clocks you want to use, and specify the clock name and output divider. The software calculates and displays the resulting output frequency.

Chapter 9

Advanced PLL Interface

Contents:

The following sections describe the advanced PLL and how to use it. Refer to the Package/Interface Support Matrix on page 8 to find out if your FPGA supports the advanced PLL.

About the Advanced PLL Interface

Trion FPGAs have PLLs to synthesize clock frequencies. The number of PLLs depends on the FPGA and package.

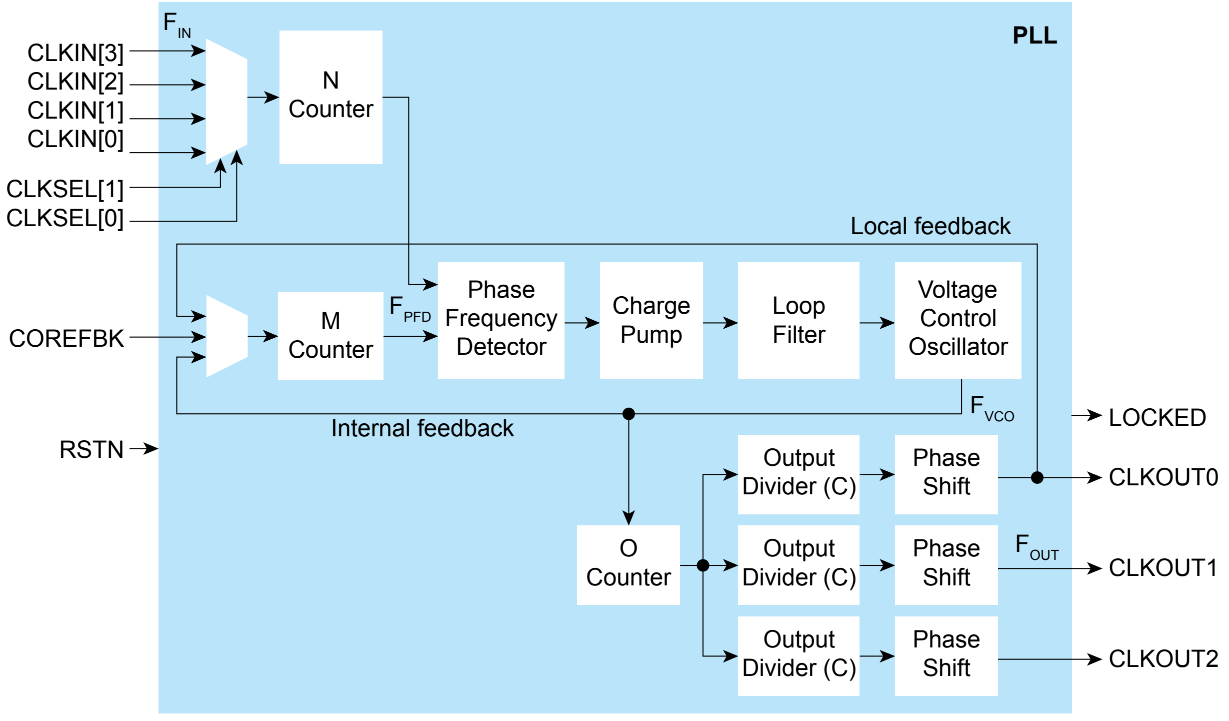
You can use the PLL to compensate for clock skew/delay via external or internal feedback to meet timing requirements in advanced application. The PLL reference clock has up to four sources. You can dynamically select the PLL reference clock with the `CLKSEL` port. (Hold the PLL in reset when dynamically selecting the reference clock source.)

Depending on the package, one or more of the PLLs can use an LVDS RX buffer to input it's reference clock.

The PLL consists of a pre-divider counter (N counter), a feedback multiplier counter (M counter), a post-divider counter (O counter), and output divider.

Note: Refer to Interface Floorplans on page 83 for the location of the PLLs on the die.

Figure 33: Advanced PLL Block Diagram

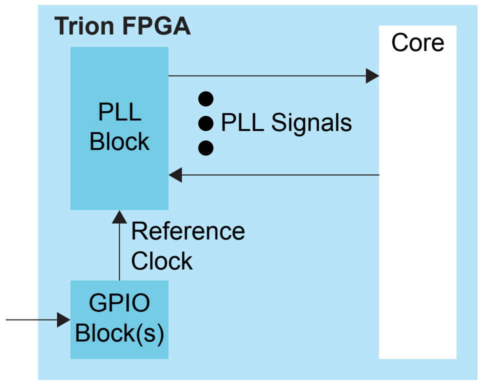


The counter settings define the PLL output frequency:

Internal Feedback Mode	Local and Core Feedback Mode	Where:
$F_{PFD} = F_{IN} / N$ $F_{VCO} = F_{PFD} \times M$ $F_{OUT} = (F_{IN} \times M) / (N \times O \times C)$	$F_{PFD} = F_{IN} / N$ $F_{VCO} = (F_{PFD} \times M \times O \times C_{FBK})$ [10] $F_{OUT} = (F_{IN} \times M \times C_{FBK}) / (N \times C)$	F_{VCO} is the voltage control oscillator frequency F_{OUT} is the output clock frequency F_{IN} is the reference clock frequency F_{PFD} is the phase frequency detector input frequency C is the output divider

Note: The reference clock input must be within the values stated in the PLL Timing and AC Characteristic section of your Trion® FPGA.

Figure 34: Advanced PLL Interface Block Diagram



[10] $(M \times O \times C_{FBK})$ must be ≤ 255.

Table 91: Advanced PLL Signals (Interface to FPGA Fabric)

Signal	Direction	Description
CLKIN[3:0]	Input	Reference clocks driven by I/O pads or core clock tree.
CLKSEL[1:0]	Input	You can dynamically select the reference clock from one of the clock in pins.
RSTN	Input	Active-low PLL reset signal. When asserted, this signal resets the PLL; when de-asserted, it enables the PLL. Connect this signal in your design to power up or reset the PLL. Assert the RSTN pin for a minimum pulse of 10 ns to reset the PLL. Assert RSTN when dynamically changing the selected PLL reference clock.
COREFBK	Input	Connect to a clock out interface pin when the the PLL feedback mode is set to core.
CLKOUT0 CLKOUT1 CLKOUT2	Output	PLL output. The designer can route these signals as input clocks to the core's GCLK network.
LOCKED	Output	Goes high when PLL achieves lock; goes low when a loss of lock is detected. Connect this signal in your design to monitor the lock status.

Table 92: Advanced PLL Interface Designer Settings - Properties Tab

Parameter	Choices	Notes
Instance Name	User defined	
PLL Resource		The resource listing depends on the FPGA you choose.
Clock Source	External	PLL reference clock comes from an external pin.
	Dynamic	PLL reference clock comes from an external pin or the core, and is controlled by the clock select bus.
	Core	PLL reference clock comes from the core.
Automated Clock Calculation		Pressing this button launches the PLL Clock Caclulation window. The calculator helps you define PLL settings in an easy-to-use graphical interface.

Table 93: Advanced PLL Interface Designer Settings - Manual Configuration Tab

Parameter	Choices	Notes
Reset Pin Name	User defined	
Locked Pin Name	User defined	
Feedback Mode	Internal	PLL feedback is internal to the PLL resulting in no known phase relationship between clock in and clock out.
	Local	PLL feedback is local to the PLL. Aligns the clock out phase with clock in.
	Core	PLL feedback is from the core. The feedback clock is defined by the COREFBK connection, and must be one of the three PLL output clocks. Aligns the clock out phase with clock in and removes the core clock delay.
Reference clock Frequency (MHz)	User defined	
Multiplier (M)	1 - 255 (integer)	M counter.
Pre Divider (N)	1 - 15 (integer)	N counter.
Post Divider (O)	1, 2, 4, 8	O counter.
Clock 0, Clock 1, Clock 2	On, off	Use these checkboxes to enable or disable clock 0, 1, and 2.
Pin Name	User defined	Specify the pin name for clock 0, 1, or 2.
Divider (C)	1 to 256	Output divider.
Phase Shift (Degree)	0, 45, 90, 135, 180, or 270	Phase shift CLKOUT by 0, 45, 90, 135, 180, or 270 degrees. 180, and 270 require the C divider to be 2. 45 and 135 require the C divider to be 4. 90 requires the C divider to be 2 or 4. To phase shift 225 degrees, select 45 and invert the clock at the destination. To phase shift 315 degrees, select 135 and invert the clock at the destination.
Use as Feedback	On, off	

Using the PLL V2 Block

Trion FPGAs (except the T4 and T8 in BGA49 and BGA81 packages) have an advanced PLL. This PLL is referenced as V2. This block lets you configure the reference clock, feedback options, frequency, and output clocks for the PLL. You can set up the PLL using the PLL Clock Calculator or manually using the Block Editor.

- In the PLL's **Properties** tab, you specify general settings such as the instance name, PLL resource, clock source, and external clock.
- Click the **Automated Clock Calculation** button to open the PLL Clock Calculator.
- Click the **Manual Configuration** tab to configure the PLL manually.

Note: For FPGAs with DDR, PLL_BR0 is the clock resource for the DDR block. If you are using the DDR block with PLL_BR0, the PLL's CLKOUT0 can only drive the DDR PHY. You *can* use the PLL's CLKOUT1 and CLKOUT2 while the DDR is using CLKOUT0.

Reference Clock Settings

The PLL has four possible reference clocks. Two of the clocks can come from the FPGA core, and two can come from off chip. You select the clocks using the **Clock Source** drop-down box:

- **core**—The PLL reference clock comes from the FPGA core.
- **external**—Enables clock `0` and clock 1. The PLL reference clock comes from an external pin. The GUI displays the resource(s) that can be the reference clock.

  Note: In this mode, a GPIO or LVDS RX block with a **pll_clkin** connection type must generate the reference clock(s). The software displays which resource you need to use (and the instance name if you have created it).

1. Add a GPIO block.
2. Enter the instance name.
3. Choose **input** as the mode.
4. Choose **pll_clkin** as the connection type.
5. In the Resource Assigner, assign it to the resource shown in the PLL's Properties tab.

- **dynamic**—Enables all four clocks; requires a clock selector bus to choose the clock dynamically. The GUI displays the resource(s) that can be the reference clock.

Using the PLL Clock Calculator

The PLL Clock Calculator provides a graphical way for you to set up the advanced PLL block. When you open the calculator, the GUI appears in automatic mode, which provides basic settings. You can:

- Choose the feedback mode (internal, core or local).
- Turn signals on (gray x) or off (green arrow) by clicking the icons next to the signal.
- Specify the signal names.
- Choose the clock phase.
- Choose which clock has feedback (for core feedback mode).

As you make selections, the calculator determines the values for the pre-divider, multiplier, post divider, and clock dividers that meet your settings. The GUI prompts you if you make selections that are impossible to solve.

In manual mode, the interface displays the PLL's internal block diagram, and provides boxes for you to set the values for the pre-divider, multiplier, post divider, and clock dividers. As you adjust the values, the calculator prompts you if you make settings that result in F_{VCO} values that are out of range or are impossible to solve. When you turn manual mode off, the calculator adjusts the output clock frequencies to match the manual settings. If you have incorrect settings for the pre-divider, multiplier, post divider, and clock dividers, when you turn manual mode off, the calculator adjusts the values to ones that allow a valid solution.

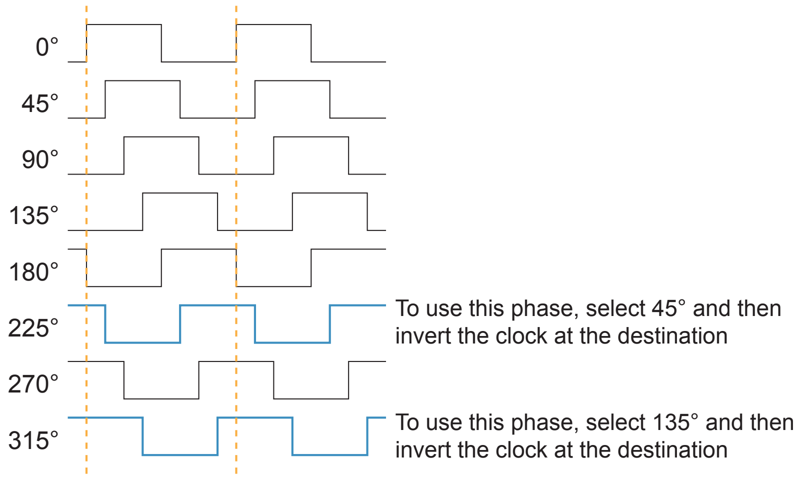
When you are finished using the calculator, click **Finish** to save your settings and close the GUI.

Understanding PLL Phase Shifting

The PLL supports clock phases from 0 to 315 degrees.

- You can select phases 0, 45, 90, 135, 180, and 270 in the Interface Designer directly.
- For phase 225, select 45 in the Interface Designer and then invert the clock at the destination.
- For phase 315, select 135 in the Interface Designer and then invert the clock at the destination.

Figure 35: PLL Clock Phases


0°
45°
90°
135°
180°
225°
270°
315°
To use this phase, select 45° and then invert the clock at the destination
To use this phase, select 135° and then invert the clock at the destination

Invert the Clock in the Interface Designer

If you connect the PLL clock output to a GPIO and want to invert it at the GPIO, use the Interface Designer GPIO Block Editor to do the inversion:

1. Add the GPIO block.
2. Choose **clkout** as the **Mode**.
3. Turn on the **Inverted** option.

Invert the Clock in Verilog HDL

This Verilog HDL example shows how to invert the clock `clk_45`:

```
always @ (negedge clk_45) begin  // the negative edge inverts the clock
    <your code>
end
```

Invert the Clock in VHDL

This VHDL example shows how to invert the clock `clk_45`:

```
process (clk_45)
    begin  -- process
      if falling_edge(clk_45) then  // the falling edge inverts the clock
        <your code>
      end if;
end process;
```

Configuring the PLL Manually

If you do not want to use the PLL clock calculator, you can manually configure the PLL using the **Manual Configuration** tab.

Specify the reset and locked pin names. If you do not want to use them, leave the boxes empty.

Choose the feedback mode. The PLL supports these modes:

- **core**—The PLL feedback comes from the FPGA core. The feedback clock must be one of the three PLL output clocks. The output clock and reference clock phases are aligned, and the is no core clock delay.Turn on the **Use as feedback** option for the clock you want to use for feedback.
- **internal**—The PLL feedback is internal to the PLL. The reference clock(s) and output clock(s) have no phase relationship.
- **local**—The PLL uses clock `0`. The feedback is local to the PLL and the output clock is aligned with the reference clock.

Note: In local and core feedback modes, the post-divider and output divider of the clock used for feedback affect the VCO frequency.

Specify the reference clock frequency, multiplier, and pre-divider. The software calculates and displays the resulting VCO frequency. If the VCO is outside of the allowed range, it displays in red.

Choose the post divider. The software calculates and displays the PLL frequency.

The advanced PLL has three output clocks. Enable the output clocks you want to use, and specify the pin name, phase shift, and output divider and whether to use the clock as feedback (core mode only). The software calculates and displays the resulting output frequency.

Output Clock Swapping

When you perform a design check or generate constraints, the software tries to find a legal routing for the PLL output clock (`clkout0`, `clkout1`, or `clkout2`). To create a legal routing, it may swap the clock output setting (for example, `clkout0` to `clkout1` or vice versa). When this swap happens, the software updates the PLL block to reflect the change. The original naming is preserved and the result is functionally equivalent.

Chapter 10

Oscillator Interface

Contents:

Oscillator

The Trion has 1 low-frequency oscillator tailored for low-power operation. The oscillator runs at nominal frequency of 10 kHz. Designers can use the oscillator to perform always-on functions with the lowest power possible. Its output clock is available to the GCLK network.

Using the Oscillator Block

To use the oscillator, specify the instance name. Choose the resource and the clock pin name.

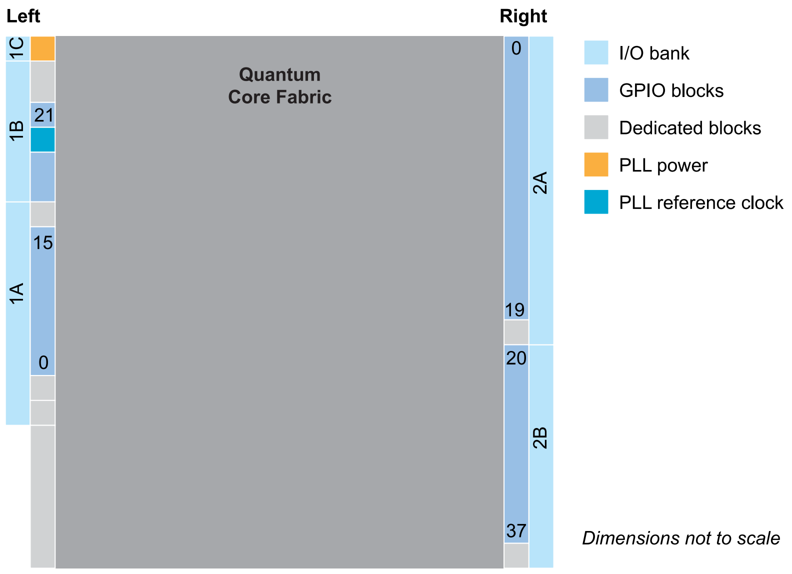
Note: You can disable the internal oscillator in Trion FPGAs. The internal oscillator is disabled if it is not instantiated in the Efinity® Interface Designer.

Chapter 11

Interface Floorplans

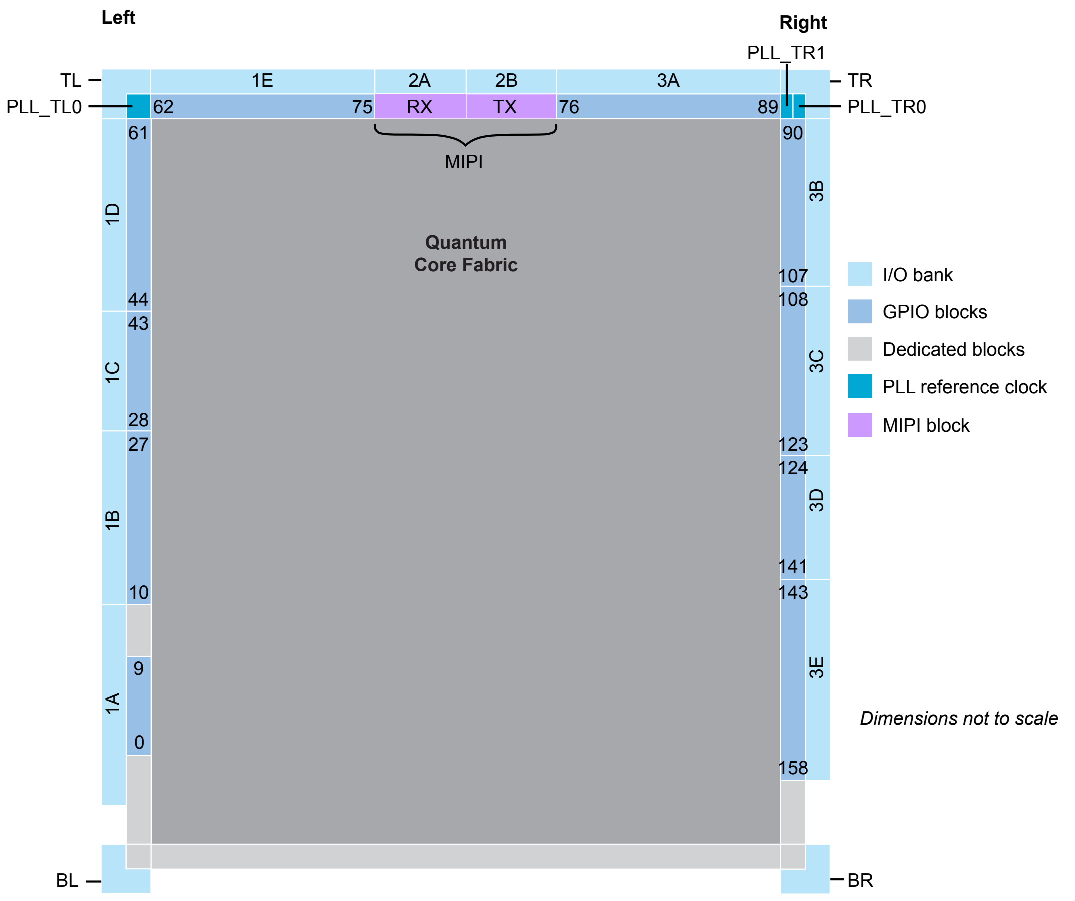
Floorplan Diagram for FPGAs in BGA49 and BGA81 Packages

Figure 36: T4 and T8 FPGAs



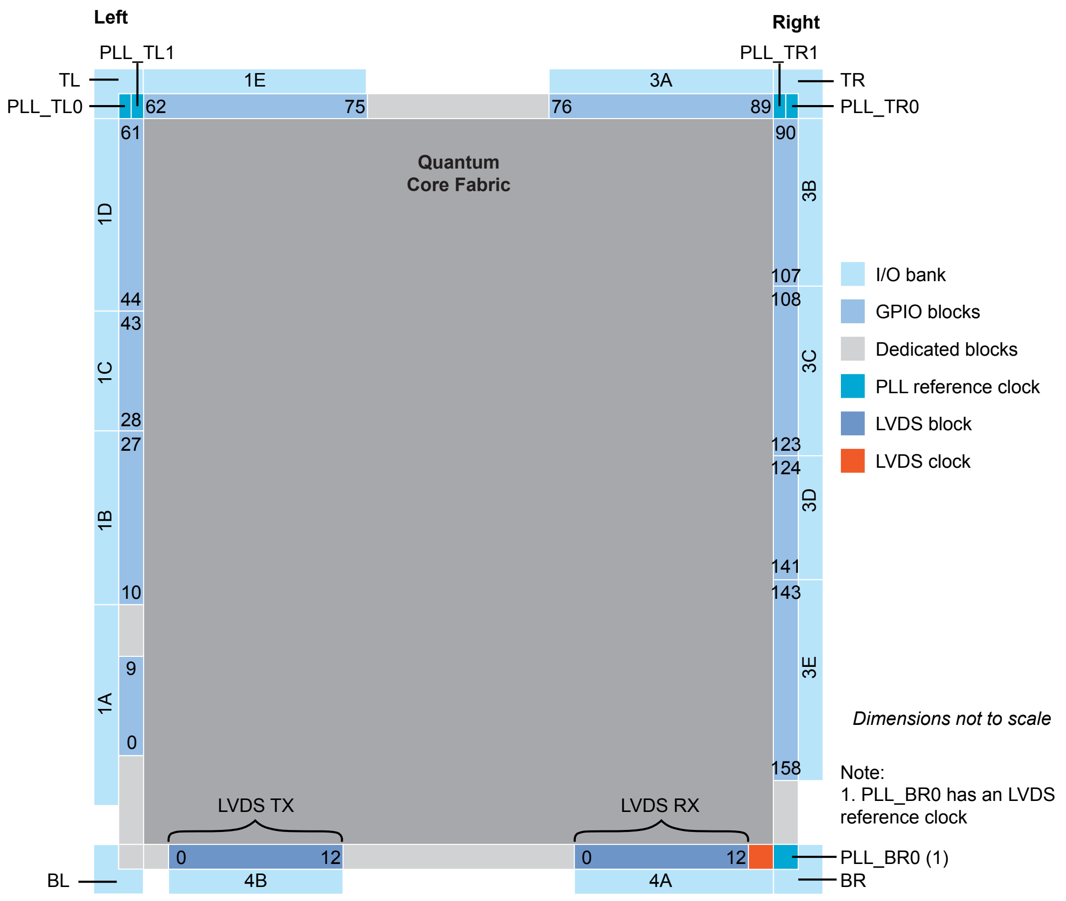
Floorplan Diagram for FPGAs in WLCSP80 Packages

Figure 37: T20 FPGAs



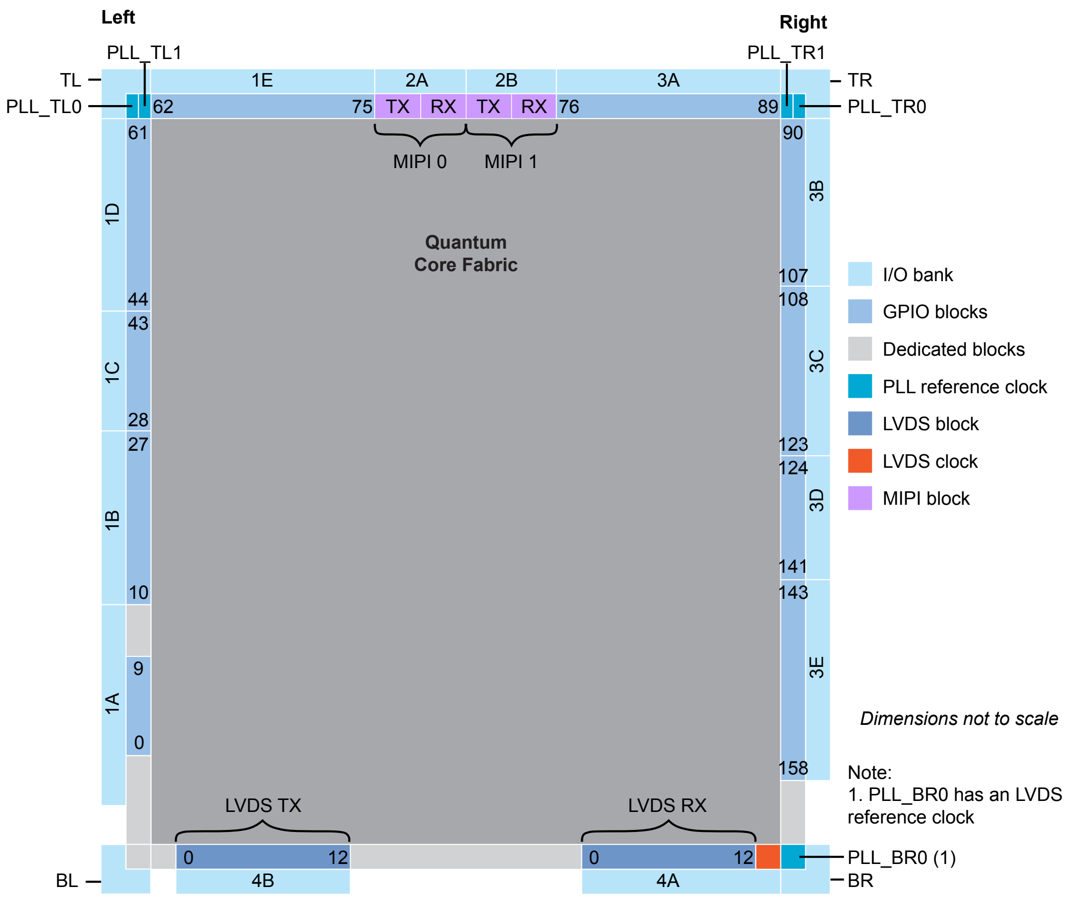
Floorplan Diagram for FPGAs in QFP144 Packages

Figure 38: T8 FPGAs



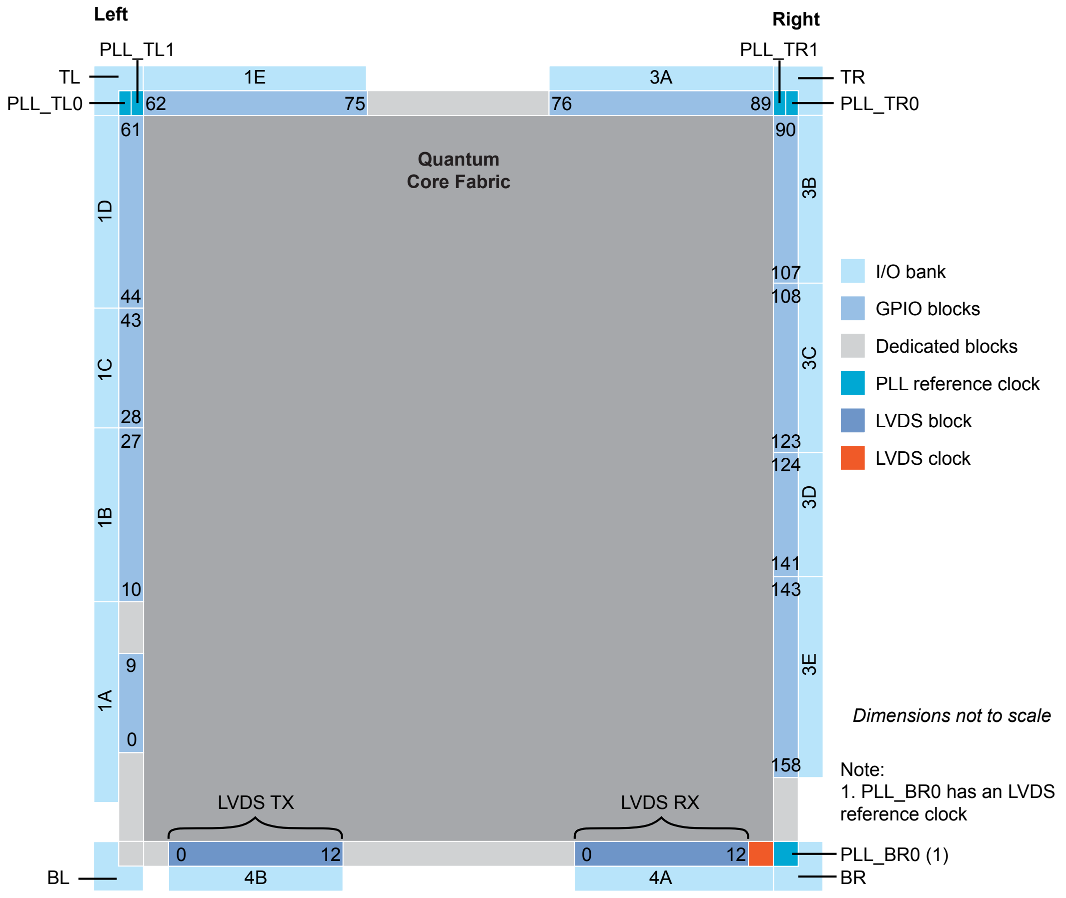
Floorplan Diagram for FPGAs in BGA169 Packages (with MIPI)

Figure 39: T13 and T20 FPGAs



Floorplan Diagram for FPGAs in BGA256 Packages

Figure 40: T13 and T20 FPGAs



Floorplan Diagrams for FPGAs in BGA324 Packages (with DDR and MIPI)

Figure 41: T20 and T35 FPGAs

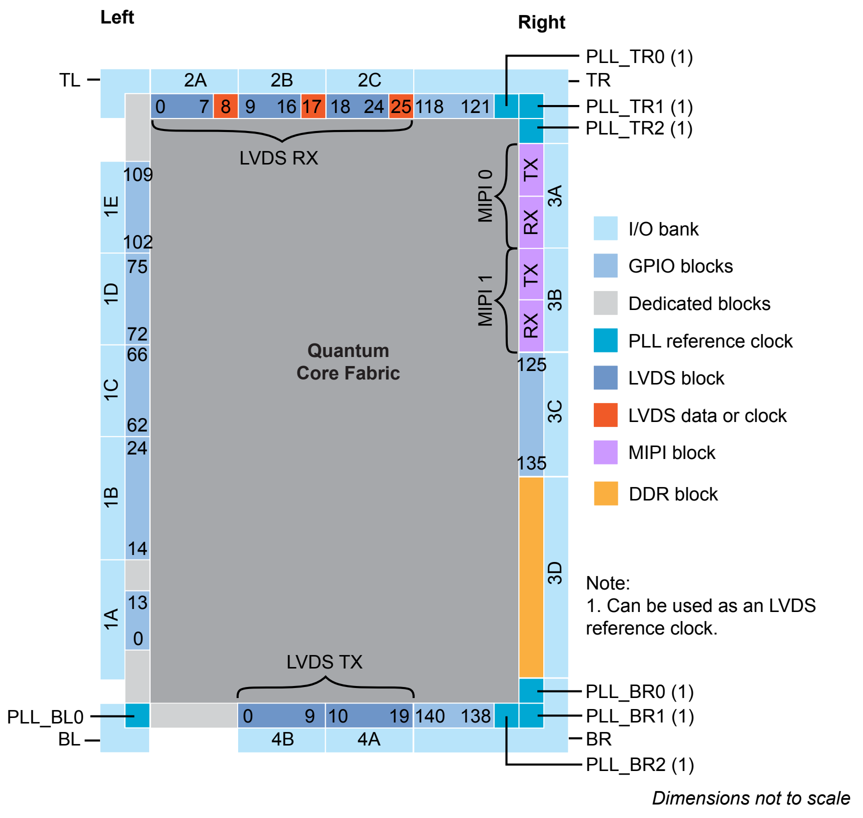
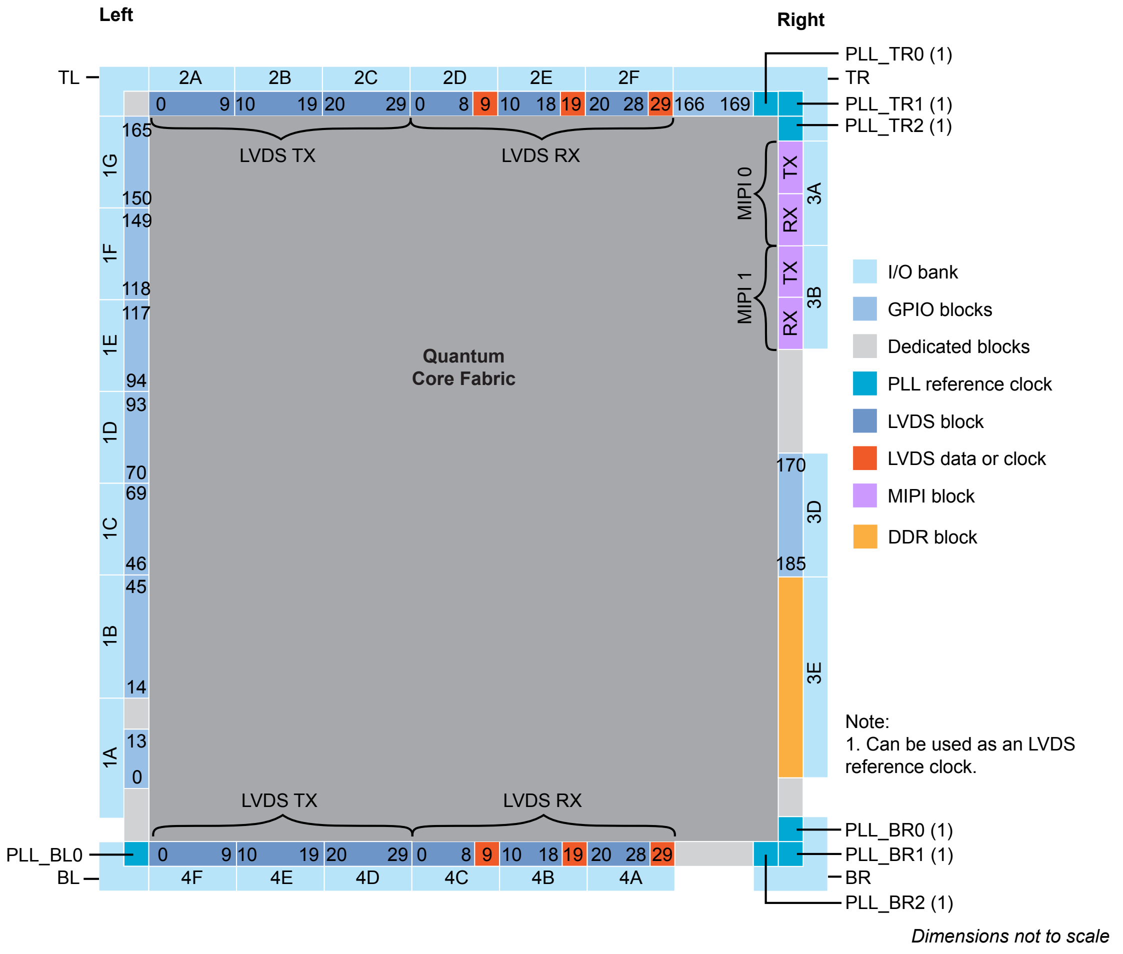
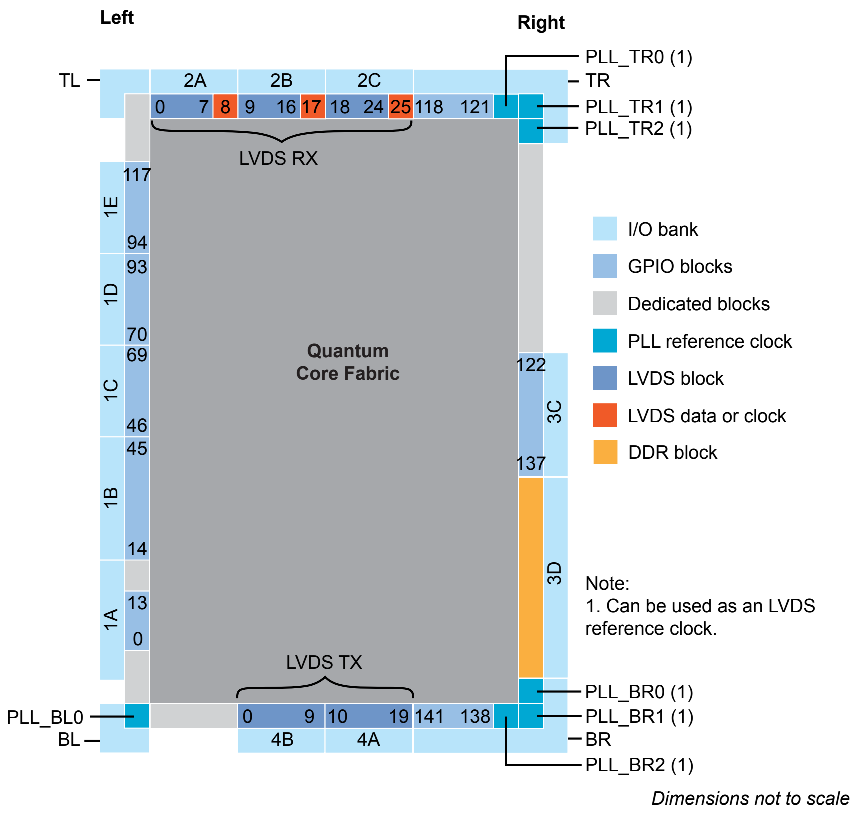


Figure 42: T55, T85, T120



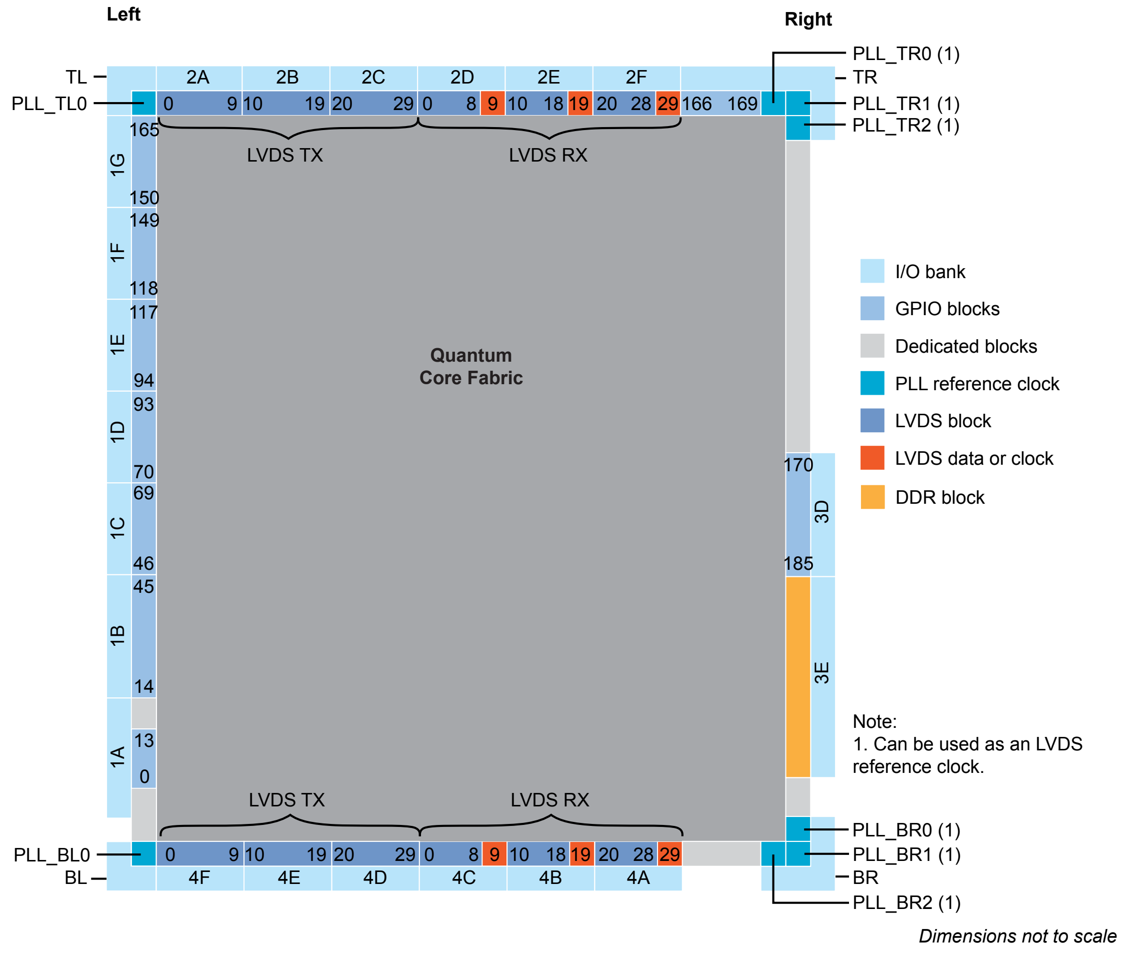
Floorplan Diagram for FPGAs in BGA400 Packages (with DDR)

Figure 43: T20 and T35 FPGAs



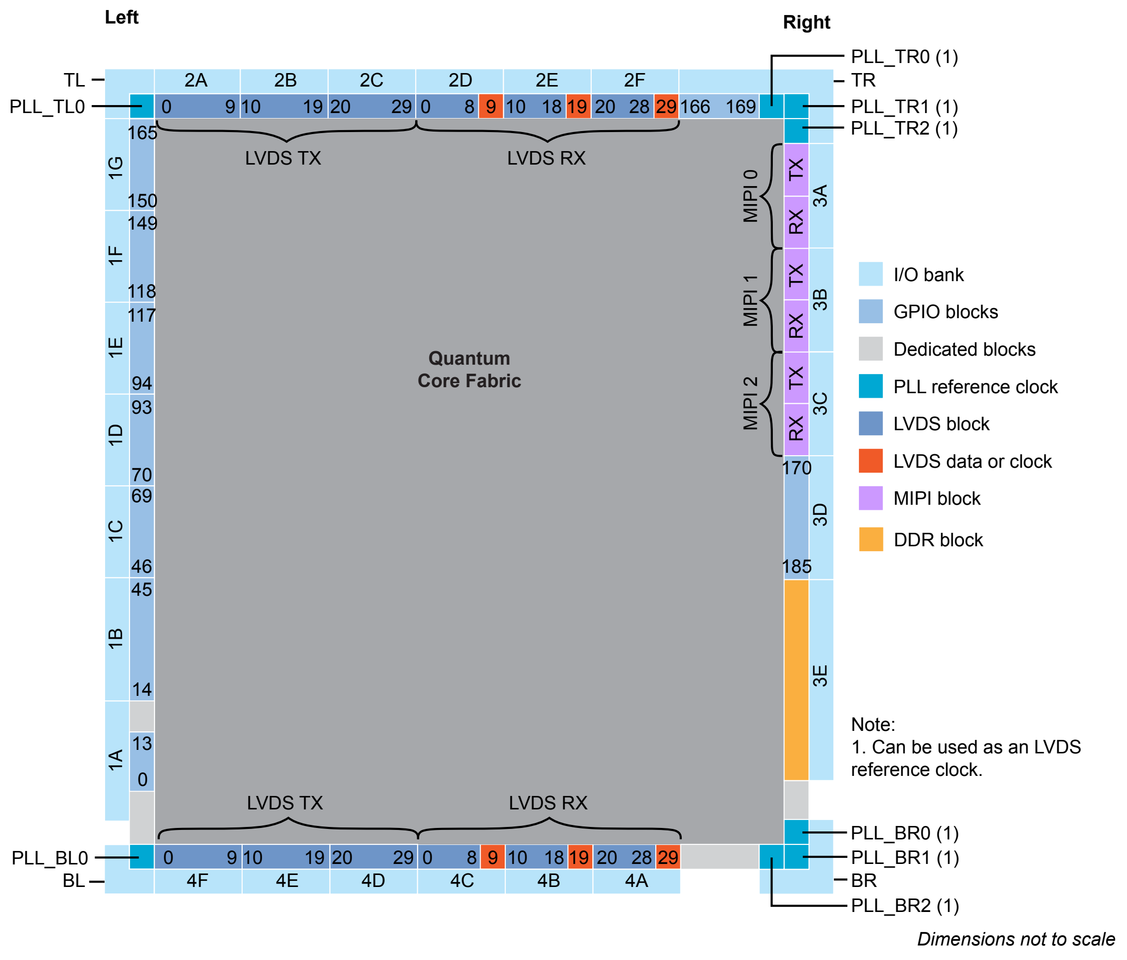
Floorplan Diagram for FPGAs in BGA484 Packages (with DDR)

Figure 44: T55, T85, and T120 FPGAs



Floorplan Diagram for FPGAs in BGA576 Packages (with DDR and MIPI)

Figure 45: T55, T85, and T120



Icon Reference

Interface Designer Icons

- Interface Designer
- Add Block
- Create a GPIO bus
- Delete Block
- Show or Hide Block Editor
- Export GPIO Assignments
- Import GPIO Assignments
- Check Design for Errors
- Export Settings
- Generate Constraints File
- View Report

Resource Assigner

- Toggle Instance View and Resource View
- Clear Resource
- Clear All Resources
- Show/Hide Filter
- Clear Filter

Revision History

Table 94: Revision History

Date	Version	Description
June 2021	7.4	Updated recommendation for PLL settings for the LVDS clock signal. (DOC-467) Renamed simple PLL as V1 and renamed advanced PLL as V2.
February 2021	7.3	Removed TX and RX timing example for serialization width of 7 and added LVDS TX data and clock relationship waveform for width 8 and 7. Added Parallel Clock Division parameter to the LVDS TX Interface Designer settings.
February 2021	7.2	Added note in Oscillator Interface stating that the oscillator is disabled if not instantiated in Interface Designer. (DOC-370) Updated Density parameter description and added 256Mb to choice to LPDDR2 in DDR Interface Designer Settings. (DOC-377) Added LVDS TX and RX timing example for serialization width of 7. (DOC-359)
December 2020	7.1	Removed RST from LVDS RX and TX interface diagrams as they are not supported in software. (DOC-362)
December 2020	7.0	Update MIPI TX and RX Interface Block Diagram to include signal names. Updated REF_CLK description for clarity. Added notes to Output Load parameter in LVDS TX Settings in Efinity® Interface Designer table. Changed the name of the GPIO connection type from none to normal. Some alternate connection types are available as inputs to the core. Described how to use the PLL calculator for the simple PLL. Updated the notes for Output Load parameter in LVDS TX Settings in Efinity Interface Designer. (DOC-309) Updated PLL reference clock input note by asking reader to refer to PLL Timing and AC Characteristics. (DOC-336) Removed OE and RST from LVDS block as they are not supported in software. (DOC-328) Added floorplan diagram for T20 FPGAs in WLCSP80 package. Added WLCSP80 package to the Package/Interface matrix.

Date	Version	Description
July 2020	6.0	Added supported features for GPIO and LVDS as GPIO. Added a topic on using LVDS as GPIO. Added note to LVDS RX interface block diagram. Added note about using output divider value of 4 when the LVDS receiver speed is higher 600 Mbps. Updated the LVDS RX and TX serilization and alternate function option descriptions. Updated the maximum F_{VCO} for advanced PLL to 1,600 MHz. You can use the PLL's CLKOUT1 and CLKOUT2 while the DDR is using CLKOUT0. The DDR PLL reference clock must be driven by I/O pads. Updated the DDR DRAM reset signal from RST_N to CFG_RST_N. Corrected DDR DRAM block diagram by adding DDR_CK signal. In MIPI RX and RTX interface description, updated maximum data pixels for RAW10 data type. Removed all instances of DDR3U. Added note to refer to AN 021 for boundary-scan testing information. Removed Efinity Interface Designer JTAG User TAP Interface subsection and added note and link to Efinity® Software User Guide for more information about JTAG User TAP interface. Added BGA400 package to interface matrix. Added BGA400 interface diagram. Added BGA400 I/O bank information.
July 2020	6.0	Updated PLLCLK pin name to PLL_CLKIN. Added PLL_EXTFB and MIPI_CLKIN as an alternative input in GPIO signals table for complex I/O buffer. Updated Memory CAS Latency (CL) choices in Advanced Options tab - Memory Mode register settings subtab. Updated Output Drive Strength choices for LPDDR2 in Advanced Options tab - Memory Mode register settings subtab. Corrected Enable Target 1 parameter notes in AXI 0 and AXI 1 tabs. Removed restriction on CLKOUT1 and CLKOUT2 when CLKIN is used to drive the DDR on CLKOUT0 in DDR DRAM PHY signals table. Updated description of how to select double data I/O for GPIO blocks.
December 2019	5.0	Enhanced the DDR interface description. Added a note about restrictions when using PLL_BR0 with the DDR block. Explained how to change the state of unused GPIO (pull up or down).
August 2019	4.0	Enhanced the MIPI interface description. Described the enhanced Resource Assigner. Added new FPGA and package support. Restructured the I/O bank information into a table. Clarified voltage support for DDR I/O banks.
April 2019	3.0	Added information for T55, T85, and T120 FPGAs. Updated the MIPI interface description. Added the DDR interface description.
January 2019	2.0	Added JTAG User TAP interface description. Added DDIO information to GPIO section. Published content in PDF as well as HTML Help. Minor changes throughout.

Date	Version	Description
October 2018	1.0	Initial release.